

Original document

COMPUTER SYSTEM AND DATA SAVING METHOD

Publication number: JP11120055

Publication date: 1999-04-30

Inventor: SEKIDO KAZUNORI

Applicant: TOKYO SHIBAURA ELECTRIC CO

Classification:


- international: **G06F12/00; G06F17/30; G06F12/00; G06F17/30; (IPC1-7): G06F12/00**

- European:

Application number: JP19970279157 19971013

Priority number(s): JP19970279157 19971013

Also published:

 US6311193

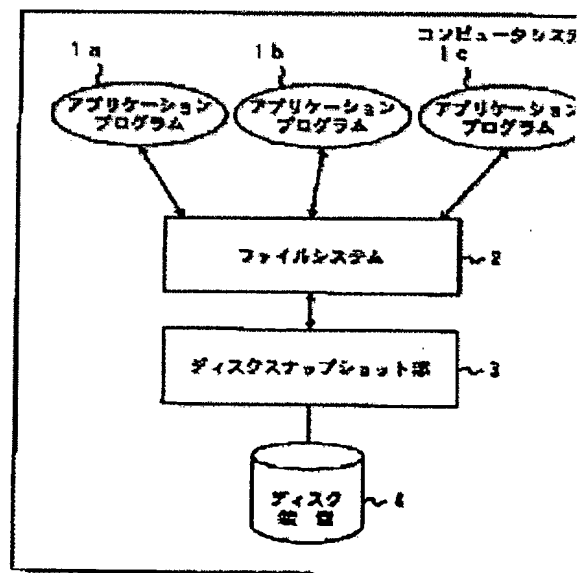
[View INPADOC patent family](#)

[View list of citing documents](#)

[Report a data error](#)

Abstract of JP11120055

PROBLEM TO BE SOLVED: To provide the computer system which efficiently gathers a snapshot for restoring the contents of a file to the contents at a specific point without modifying an existing operating system, a file system, an application program, etc. **SOLUTION:** Between a file system 2 which manages a file group that application programs 1a to 1c again access to including update and more than one disk drive 4 stored with the file group, a disk snapshot part 3 is interposed which gathers snapshot information for restoring the contents of the file group to the contents at a specific point of time by the disk drives 4 and stores the gathered snapshot information on the respective disk drivers 4. Consequently, snapshots can efficiently be gathered without modifying the application programs 1a to 1c, file system 2, etc.



Data supplied from the *esp@cenet* database - Worldwide

Description of corresponding document: US6311193

Translate this text

BACKGROUND OF THE INVENTION

This invention relates to a computer system which efficiently creates snapshots holding the contents specific point in time of the files stored in a nonvolatile storage device.

A redundant disk structure has been used to cope with defects in a single disk. The redundancy technique, however, is useless against the loss or change of programs or data due to software bugs, erroneous operations, or infection with viruses, or against such a serious fault as the loss of a disk storage device itself. To deal with these problems, it is essential to create snapshots or backups.

A snapshot is a copy image of a file or a disk at a certain point in time. A file or the whole disk is copied at regular intervals into the same disk storage device or a different disk storage device to create a snapshot. In case a program or data has been lost or changed, the fault will be coped with by restoring the program or data or the whole disk in the snapshot created immediately before the occurrence of the fault.

A backup is a copy of the snapshot saved and stored in a different mass storage system (e.g., magnet tapes). With backups, even if the disk storage device itself has been lost (and naturally the whole data device has been lost), the programs or data at the time of creating the snapshots can be restored by loading the backups into a new disk storage device.

In general, to create snapshots or backups, it is necessary to stop all of the application programs that possibly change the related files, or else they would change the files or data in the middle of creating a copy, preventing the snapshot or backup from being created correctly. Because restoring the incorrect snapshot or backup to the disk storage device introduces the danger of permitting an error or a faulty operation to occur on the application program side, the approach cannot be used to deal with the fault.

Since data is generally written into a disk storage device faster than into a mass storage system, creating snapshots and backing up them on a mass storage medium shortens the stopping time of application programs more than backing up the files or disk images directly. Use of snapshots enables the program data from being recovered from the loss or change in a shorter time. When backups are made using a sequential access medium, such as magnetic tape, a lot of time is needed to find a specific file, resulting in a very low efficiency.

File systems have been disclosed which have a snapshot function that solves the problem of having to stop the application programs for a long time to create snapshots or backups (reference 1: "The Episode File System," Proceedings of the Winter 1992 USENIX Conference, pp. 43-60, San Francisco, Calif.; reference 2: "File System Design for an NFS File Server Appliance," Proceedings of the Winter 1994 USENIX Conference, pp. 235-244, San Francisco, Calif.).

These file systems (e.g., in reference 2) are based on the assumption that a file has a block tree structure with root I node as the starting point as shown in FIG. 1(a). At the time of creating a snapshot, a copy of the root I node is made. The copied I node (snapshot I node) represents a block tree structure as a snapshot. The root I node to be copied represents a block tree structure of the active file system which data is read from and written into. Because the snapshot I node points at the same block as the root I node when the snapshot is just been created (FIG. 1(b)), the disk space is not used at all, except that it is used for the copied I node in a new snapshot.

Suppose the user has changed data block D. In this case, as shown in FIG. 1(c), a new data block D' is written onto a disk and the file system represented by root I node is changed to point at the new block D'. The original data block D on the disk remains unchanged and snapshot I node still points at the original data block D.

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平11-120055

(43) 公開日 平成11年(1999) 4月30日

(51) Int.Cl.⁶

G 0 6 F 12/00

識別記号

5 3 1

F I

C 0 6 F 12/00

5 3 1 J

審査請求 未請求 請求項の数31 O L (全 25 頁)

(21) 出願番号 特願平9-279157

(22) 出願日 平成9年(1997)10月13日

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 関戸 一紀

東京都青梅市末広町2丁目9番地 株式会
社東芝青梅工場内

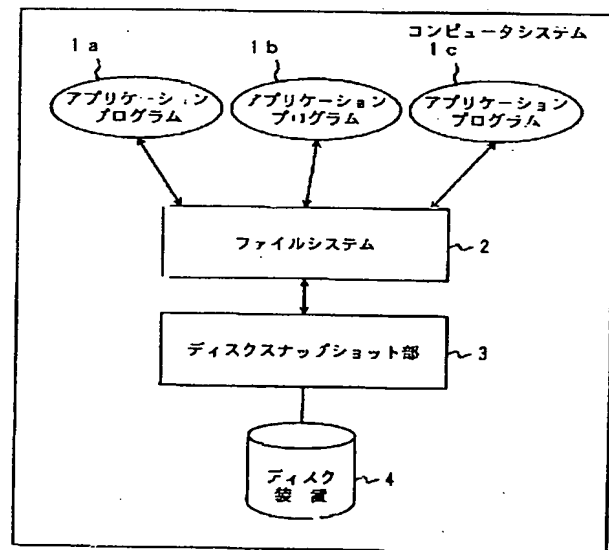
(74) 代理人 弁理士 鈴江 武彦 (外6名)

(54) 【発明の名称】 コンピュータシステムおよびデータ退避方法

(57) 【要約】

【課題】 既存のオペレーティングシステム、ファイルシステムおよびアプリケーションプログラムなどの改造を伴わずにファイルの内容を所定の時点に復元するためのスナップショットを効率的に採取するコンピュータシステム。

【解決手段】 アプリケーションプログラム1a~1cが更新を含むアクセスを実行するファイル群を管理するファイルシステム2と、このファイル群を格納する少なくとも一つ以上のディスク装置4との間に、ファイル群の内容を所定の時点に復元するためのスナップショット情報をディスク装置4ごとに採取し、この採取したスナップショット情報をそれぞれのディスク装置4に格納するディスクスナップショット部3を介在させて設けることにより、アプリケーションプログラム1a~1cやファイルシステム2などになんらの変更を伴わずにスナップショットを効率的に採取することを可能とする。



【特許請求の範囲】

【請求項1】 不揮発性記憶装置に格納されたファイル群に対するアプリケーションプログラムからの更新を含むアクセスを管理するファイルシステムを備えたコンピュータシステムにおいて、

前記ファイルシステムと前記不揮発性記憶装置との間に介在し、前記ファイル群の内容を所定の時点に復元するためのスナップショットを前記不揮発性記憶装置ごとに採取して、それぞれの不揮発性記憶装置上に格納するスナップショット管理手段を具備することを特徴とするコンピュータシステム。

【請求項2】 前記スナップショットに対するアクセスを管理するスナップショット用ファイルシステムをさらに具備し、

前記スナップショット管理手段は、前記スナップショット用ファイルシステムに対して前記不揮発性記憶装置を別の擬似記憶装置に見せ掛ける手段を有することを特徴とする請求項1記載のコンピュータシステム。

【請求項3】 前記スナップショット管理手段は、前記スナップショット用ファイルシステムに対して前記擬似記憶装置をメディアが差し替え可能なリムーバブル記憶装置に見せ掛ける手段を有することを特徴とする請求項2記載のコンピュータシステム。

【請求項4】 前記スナップショットに対するアクセスを管理するスナップショット用ファイルシステムをさらに具備し、

前記スナップショット管理手段は、前記ファイルシステムおよび前記スナップショット用ファイルシステムに対して前記不揮発性記憶装置をメディアが差し替え可能なリムーバブル記憶装置に見せ掛ける手段を有することを特徴とする請求項1記載のコンピュータシステム。

【請求項5】 前記スナップショット管理手段は、システム起動時に選択されたスナップショットの採取時の内容を前記ファイル群が示すように前記不揮発性記憶装置を立ち上げる手段を有することを特徴とする請求項1、2、3または4記載のコンピュータシステム。

【請求項6】 前記スナップショット管理手段は、次のシステム起動時に選択すべきスナップショットを記録する手段を有することを特徴とする請求項5記載のコンピュータシステム。

【請求項7】 前記スナップショット管理手段は、前記不揮発性記憶装置上に格納された前記スナップショットに対して追加的な変更を施す手段を有することを特徴とする請求項1記載のコンピュータシステム。

【請求項8】 前記スナップショット管理手段は、前記不揮発性記憶装置上に格納された前記スナップショットに対して施された追加的な変更を無効とする手段を有することを特徴とする請求項7記載のコンピュータシステム。

【請求項9】 前記スナップショット管理手段は、前記

スナップショットを採取したときの日時をスナップショット情報として記録する手段を有することを特徴とする請求項1、2、3、4、5、6、7または8記載のコンピュータシステム。

【請求項10】 前記ファイルシステムは、更新データを一時的に格納するキャッシュ領域を備え、

前記キャッシュ領域に格納された更新データすべてを前記不揮発性記憶装置に反映させるように前記ファイルシステムに指示するとともに、この更新データを含むスナップショットを採取するように前記スナップショット管理手段に指示する全書き戻し状態実現手段をさらに具備することを特徴とする請求項1、2、3、4、5、6、7、8または9記載のコンピュータシステム。

【請求項11】 前記全書き戻し状態実現手段は、前記スナップショット管理手段に対して第1の指示を与えるとともに、前記ファイルシステムに対して前記更新データの前記不揮発性記憶装置への反映を指示した後、前記ファイルシステムから反映完了の返答を受け取ったときに、前記スナップショット手段に対して第2の指示を与え、

前記スナップショット管理手段は、前記第1の指示を受け取ってから第2の指示を受け取るまで間に、前記ファイルシステムから前記更新データの書き出しがないときにのみ、前記スナップショットを採取することを特徴とする請求項10記載のコンピュータシステム。

【請求項12】 不揮発性記憶装置に格納されたファイル群に対するアプリケーションプログラムからの更新を含むアクセスを管理するファイルシステムとK個の論理ブロックに相当する容量を有する書き込みバッファとを備えたコンピュータシステムに適用されるデータ退避方法において、

通常書き込み処理では、

前記書き込みバッファに更新すべきデータの論理ブロックを蓄積し、

その蓄積した論理ブロックがK-1個に達するまでその論理ブロックのデータ更新を遅延させ、

前記書き込みバッファに蓄積された各論理ブロックに対する論理アドレスから構成される論理アドレスタグブロックを生成し、

この論理アドレスタグブロックに書き込みの時間的順序を維持するためのタイムスタンプを付加し、

K-1個のブロックに前記論理アドレスタグブロックを加えたK個の論理ブロックを一つのストライプとして前記不揮発性記憶装置上の更新されるべきデータを保持する領域とは別の空領域に連続して書き込み、

前記ファイル群の内容を所定の時点に復元するためのスナップショットを採取するスナップショット採取処理では、

スナップショット採取時点で更新すべきデータの論理ブロックが埋まっていない前記書き込みバッファ内のK-

1-L個の各論理ブロックに対する論理アドレスをマルチアドレスとし、

スナップショット採取時点の前記書き込みバッファに蓄積されたL個の各論理ブロックに対する論理アドレスと前記マルチアドレスとから構成される論理アドレスタグブロックを生成し、

この論理アドレスタグブロックに書き込みの時間的順序を維持するためのタイムスタンプを付加し、

前記L個の論理ブロックを含むK-1個のブロックに前記論理アドレスタグブロックを加えたK個の論理ブロックを一つのストライプとして前記不揮発性記憶装置上の更新されるべきデータを保持する領域とは別の空領域に連続して書き込み、

前記付加したタイムスタンプをスナップショット情報として記録することを特徴とするデータ退避方法。

【請求項13】 前記採取したスナップショットに関しては前記不揮発性記憶装置を別の擬似記憶装置に見せ掛け、

この擬似記憶装置に対する読み出し要求に対しては、前記スナップショット情報として記録されたタイムスタンプで示される時刻またはそれ以前に書き込まれたストライプであって、読み出し要求された論理アドレスのブロックを含む最も後に書き込まれたストライプを検索し、

この検索により検出されたストライプの要求された論理アドレスに対応するブロックを読み出し、その読み出したデータを要求の応答として返すことを特徴とする請求項12記載のデータ退避方法。

【請求項14】 前記採取したスナップショットを格納する前記不揮発性記憶装置を別の擬似記憶装置に見せ掛け、

前記スナップショット情報として記録されたタイムスタンプで示される時刻またはそれ以前に書き込まれたストライプであって、各々の論理アドレスのブロックを含む最も後に書き込まれたストライプに対応する物理ブロックを検索し、

前記論理アドレスと前記検索により検出された物理ブロックとの対応関係を表すスナップショット用変換マップを作成し、

前記擬似記憶装置に対する読み出し要求に対しては、前記作成したスナップショット用変換マップに基づいて、要求された論理アドレスに対応する物理ブロックを応答として返すことを特徴とする請求項12のデータ退避方法。

【請求項15】 オペレーティングシステムからの前記擬似記憶装置に関する問い合わせに対してリムーバブルであると応答を返し、スナップショット切り替え後のアクセスでメディア不確定の応答を返し、かつ切り替え指定されたスナップショットのタイムスタンプを用いて前記擬似記憶装置に対する読み出し処理を実行することを

特徴とする請求項13または14記載のデータ退避方法。

【請求項16】 システム起動時に参照すべきスナップショットを問い合わせ、

この問い合わせに応じて選択されたスナップショットのタイムスタンプを用いて前記擬似記憶装置に対する読み出し処理を実行することを特徴とする請求項13または14記載のデータ退避方法。

【請求項17】 次回のシステム起動時に選択すべきスナップショットを記録し、

この記録されたスナップショットのタイムスタンプを用いて前記擬似記憶装置に対する読み出し処理を実行することを特徴とする請求項13または14記載のデータ退避方法。

【請求項18】 前記スナップショット情報を親となるスナップショットの識別子とその後書き出されたストライプのタイムスタンプとから構成し、

スナップショットを参照する際、前記スナップショット情報で示された親となるスナップショットを辿りながら対象とするストライプを決定することを特徴とする請求項13または14記載のデータ退避方法。

【請求項19】 親となるスナップショットの識別子とその後書き出されたストライプのタイムスタンプとから最新イメージ情報を構成し、

最新イメージを読み出す際、前記最新イメージ情報で示された親となるスナップショットを辿りながら対象とするストライプを決定することを特徴とする請求項13または14記載のデータ退避方法。

【請求項20】 最新イメージに対応するタイムスタンプの範囲を予め割り当てタイムスタンプ情報として保存しておき、

最新イメージ情報のタイムスタンプとして、前記不揮発性記憶装置に書き込まれたストライプのタイムスタンプであって前記タイムスタンプ情報の範囲内にあるものを求めることを特徴とする請求項19記載のデータ退避方法。

【請求項21】 前記スナップショット情報または最新イメージ情報の削除を受け付けることを特徴とする請求項13、14、15、16、17、18、19または20記載のデータ退避方法。

【請求項22】 削除によりアクセスされなくなったストライプのタイムスタンプを削除スタンプ情報として保存する請求項13、14、15、16、17、18、19、20または21記載のデータ退避方法。

【請求項23】 前記スナップショット情報を前記タイムスタンプとストライプ内ブロック位置とにより構成し、

前記スナップショット採取後も前記書き込みバッファをクリアせず、

前記書き込みバッファ内の論理ブロックを前回の書き出

し時と同じタイムスタンプで同じストライプ上に書き出すことを特徴とする請求項13、14、15、16、17、18、19、20、21または22記載のデータ退避方法。

【請求項24】 最新イメージおよびスナップショットのいずれからも参照されないブロックを無効ブロックとして複数のストライプを新たな一つのストライプに詰め替えることを特徴とする請求項13、14、15、16、17、18、19、20、21、22または23記載のデータ退避方法。

【請求項25】 前記詰め替え後の新たなストライプのタイムスタンプを詰め替え情報として記録し、スナップショットを参照する際、前記詰め替え情報のタイムスタンプのストライプも検索対象のストライプに含めることを特徴とする請求項24記載のデータ退避方法。

【請求項26】 前記詰め替え対象となったストライプの識別子を前記詰め替え後の新たなストライプの元ストライプ情報として記録し、スナップショットを参照する際、前記元ストライプ情報を用いることを特徴とする請求項24記載のデータ退避方法。

【請求項27】 ストライプ内の各論理ブロックの状態を示すビットマップをスナップショット情報として保存し、無効ブロックの判定に用いることを特徴とする請求項24、25または26記載のデータ退避方法。

【請求項28】 無効ブロックが多いストライプのビットマップのみをスナップショット情報として保存する請求項27記載のデータ退避方法。

【請求項29】 前記スナップショット情報および最新イメージ情報に採取日時を付加し、スナップショットまたは最新イメージの識別に前記付加した日時を用いることを特徴とする請求項13、14、15、16、17、18、19、20、21、22、23、24、25、26、27または28記載のデータ退避方法。

【請求項30】 前記不揮発性記憶装置がRAID0の構成であったときには、前記ストライプサイズをストライプユニット単位×ディスク台数とする請求項12、13または14記載のデータ退避方法。

【請求項31】 前記不揮発性記憶装置がRAID3、RAID4およびRAID5のいずれかの構成であったときには、前記ストライプサイズをストライプユニット単位×(ディスク台数-1)とする請求項12、13または14記載のデータ退避方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】この発明は、たとえばディスク装置などに格納されたファイルの更新前データを後方回復のために退避するコンピュータシステムに係り、特に既存のオペレーティングシステム、ファイルシステムおよびアプリケーションプログラムなどの改造を伴なわ

ずにファイルの内容を所定の時点に復元するためのスナップショットを効率的に採取するコンピュータシステムに関する。

【0002】

【従来の技術】従来より、ディスク単体の障害に対しては、冗長化ディスク構成を採用することによって対処が可能であるが、この冗長化技術は、ソフトウェアバグ、誤操作およびウイルス感染などによるプログラムまたはデータの喪失・改変、もしくはディスク記憶装置そのものを失なうような大規模な障害には役立たない。これらの問題に対処するためには、スナップショットやバックアップなどの採取が必須となる。

【0003】スナップショットとは、ある時点のファイルやディスクなどのコピーイメージのことであり、定期的にファイルやディスク全体を同じディスク記憶装置や別のディスク記憶装置にコピーして、スナップショットを作成する。そして、プログラムまたはデータが喪失・改変された場合には、直前に作成されたスナップショット(コピーイメージ)にプログラムまたはデータ、もしくはディスク全体を戻すことによりこれらの障害に対処する。

【0004】また、バックアップとは、このスナップショットを別の大容量記憶媒体(磁気テープなど)に保存して保管したものである。これにより、ディスク記憶装置そのものを失った場合(中のデータも全部喪失)であっても、新しいディスク記憶装置にこのバックアップをロードすることによって、スナップショット採取時のプログラムまたはデータを復元することができる。

【0005】通常、スナップショットやバックアップの採取時には、それらのファイルを変更する可能性があるアプリケーションをすべて停止する必要がある。そうしないと、コピー作成中にファイルやデータが変更されてしまい、正しいスナップショットやバックアップが採取できないからである。すなわち、不正なスナップショットやバックアップをディスク記憶装置に戻しても、アプリケーションプログラム側でエラーや誤動作を発生させる危険性があるため、先の障害対策として意味をなさない。

【0006】なお、一般に大容量記憶媒体よりもディスク記憶装置への書き込みの方が高速であるので、ファイルやディスクイメージを直接バックアップするよりも、スナップショットを作成し、それを大容量記憶媒体にバックアップする方がアプリケーションの停止期間を短く押さえられる。また、プログラムまたはデータの喪失・改変からの復元もスナップショットの方が短期に行なえる。バックアップが磁気テープなどのシーケンシャルアクセスの媒体の場合、特定のファイルを見つけて読み出すには多くの時間を要してしまい非常に効率が悪い。

【0007】また、このスナップショットやバックアップを採取するためにアプリケーションを長時間停止しな

ければならないといった問題を解決する、スナップショット機能を持ったファイルシステムも開示されている(文献1:「The Episode File System」, Proceedings of the Winter 1992 USENIX Conference, pp. 43-60, San Francisco, CA、文献2:「File System Design for an NFS File Server Appliance」, Proceedings of the Winter 1994 USENIX Conference, pp. 235-244, San Francisco, CA)。

【0008】ところで、これらのファイルシステム(たとえば文献2)では、図32(a)に示すように、ファイルがルート・Iノードを起点とするブロック木構造であることを前提にしている。そして、スナップショット作成時には、このルート・Iノードのコピーを作成する。このコピーしたIノード(スナップショット・Iノード)は、スナップショットとしてのブロック木構造を表現することになる。また、コピー元のルート・Iノードは、読み出し/書き込みが行なわれるアクティブなファイルシステムのブロック木構造を表現する。スナップショット・Iノードは、生成されたときにはルート・Iノードとまったく同じブロックを指している(図32(b))、新しいスナップショットのためにコピーしたIノード以外はまったくディスクスペースを消費しない。

【0009】ここで、ユーザがデータブロックDを変更したとすると、図32(c)に示すように、新しいデータブロックD'をディスク上に書き込み、ルート・Iノードで表現するアクティブなファイルシステムが、この新しいブロックを指すように変更される。ディスク上の元のデータブロックDは変更されずに残っており、スナップショット・Iノードは、元のデータブロックDをまだ指している。したがって、スナップショット・Iノードを指定することにより、スナップショット作成時のデータブロックA、B、C、D、Eをそのまま参照できる。

【0010】このように、スナップショット機能を持ったファイルシステムを使えば、ルート・Iノードをコピーするだけで簡単にスナップショットを作成でき、アプリケーションプログラムを停止することなくスナップショットが作成できる。また、スナップショットをアプリケーションプログラムの実行と並行して大容量記憶媒体に保存できるので、バックアップの採取もアプリケーションを停止することなく行なえる。したがって、スナップショットやバックアップのデータ退避のためにアプリケーションプログラムを停止する必要が無くなる。

【0011】

【発明が解決しようとする課題】しかしながら、前述し

た手法においては、専用のファイルシステムを新しく開発する必要があり、既存のコンピュータシステムにそのまま適用できるものではなかった。また、ファイルシステム自身もブロック木構造で構成されていることを前提にしており、たとえばマイクロソフト社のNTFSなどのエクステンツベースのファイルシステムには適用できる技術ではなかった。さらに、一般にファイルシステムは大きな木構造をしており、実際に前述のデータブロックDをD'に更新するには、図33に示すように、その経路に位置する中間ノードすべてをコピーする必要があり、スナップショット作成後の更新性能が大きく低下するといった問題もあった。さらに、ファイルシステムという複雑なソフトウェアモジュールにスナップショット機能を付加したため、スナップショットは読み出しだけで非常に硬直なものになってしまっていた。

【0012】この発明はこのような実情に鑑みてなされたものであり、既存のコンピュータシステムやファイルシステムにそのまま適用することができ、かつ更新時のオーバーヘッドもなく、スナップショットの更新なども可能な非常に柔軟なデータ退避を可能とするコンピュータシステムおよびデータ退避方法を提供することを目的とする。

【0013】

【課題を解決するための手段】この発明は、前述した目的を達成するために、少なくとも一つ以上の不揮発性記憶装置に格納されたファイル群に対するアプリケーションプログラムからの更新を含むアクセスを管理するファイルシステムを備えたコンピュータシステムにおいて、前記ファイルシステムと前記不揮発性記憶装置との間に介在し、前記ファイル群の内容を所定の時点に復元するためのスナップショットを前記不揮発性記憶装置ごとに採取して、それぞれの不揮発性記憶装置上に格納するスナップショット管理手段を具備したものである。

【0014】また、この発明は、少なくとも一つ以上の不揮発性記憶装置に格納されたファイル群に対するアプリケーションプログラムからの更新を含むアクセスを管理するファイルシステムと少なくともK個の論理ブロックに相当する容量を有する書き込みバッファとを備えたコンピュータシステムに適用されるデータ退避方法において、通常書き込み処理では、前記書き込みバッファに更新すべきデータの論理ブロックを蓄積し、その蓄積した論理ブロックがK-1個に達するまでその論理ブロックのデータ更新を遅延させ、前記書き込みバッファに蓄積された各論理ブロックに対する論理アドレスから構成される論理アドレスタグブロックを生成し、この論理アドレスタグブロックに書き込みの時間的順序を維持するためのタイムスタンプを付加し、K-1個のブロックに前記論理アドレスタグブロックを加えたK個の論理ブロックを一つのストライプとして前記不揮発性記憶装置上の更新されるべきデータを保持する領域とは別の空領域

域に連続して書き込み、前記ファイル群の内容を所定の時点に復元するためのスナップショットを採取するスナップショット採取処理では、スナップショット採取時点で更新すべきデータの論理ブロックが埋まっている前記書き込みバッファ内の $K-1-L$ 個の各論理ブロックに対する論理アドレスをヌルアドレスとし、スナップショット採取時点の前記書き込みバッファに蓄積された L 個の各論理ブロックに対する論理アドレスと前記ヌルアドレスとから構成される論理アドレスタグブロックを生成し、この論理アドレスタグブロックに書き込みの時間的順序を維持するためのタイムスタンプを付加し、前記 L 個の論理ブロックを含む $K-1$ 個のブロックに前記論理アドレスタグブロックを加えた K 個の論理ブロックを一つのストライプとして前記不揮発性記憶装置上の更新されるべきデータを保持する領域とは別の空領域に連続して書き込み、前記付加したタイムスタンプをスナップショット情報として記録するようにしたものである。

【0015】すなわち、たとえばディスク装置などの不揮発性記憶装置と、この不揮発性記憶装置に格納されたファイル群に対するアプリケーションプログラムからの更新を含むアクセスを管理するファイルシステムとの間に、ファイル群の内容を所定の時点に復元するためのスナップショット情報を不揮発性記憶装置ごとに採取し、この採取したスナップショット情報をそれぞれの不揮発性記憶装置上に格納するスナップショット管理手段を設けるようにしたものである。

【0016】この発明によれば、既存のオペレーティングシステムやファイルシステム、あるいはアプリケーションプログラムになんらの改造を伴うことなく、スナップショットを効率的に採取することが可能となる。

【0017】

【発明の実施の形態】以下、図面を参照してこの発明の実施形態について説明する。

【0018】（第1実施形態）まず、この発明の第1実施形態について説明する。図1はこの実施形態に係るコンピュータシステムの構成を示す概念図である。このシステムは、スナップショットやバックアップが必要なアプリケーションプログラム1a～1c、これらのアプリケーションプログラム1a～1cが使う既存のファイルシステム2、このファイルシステム2とディスク装置4との間に位置し、ディスクレベルのスナップショットを採取するディスク・スナップショット部3およびファイルシステム2のデータやディスク・スナップショット部3のデータを保持するディスク装置4から構成される。

【0019】これらアプリケーションプログラム1a～1c、ファイルシステム2およびディスク装置4は既存のコンピュータシステムと同じ任意の要素であり、従来のようにファイルシステム2がブロック木構造である必要はまったくない。また、ファイルシステム2は既存のものそのままであり、従来のようにスナップショット機

能を持ったものである必要もない。すなわち、この発明の特徴となる構成要素はディスクスナップショット部3であるので、以下ではこのディスクスナップショット部3について説明する。

【0020】図2はこの実施形態のディスクスナップショット部3の構成を示す概念図である。図2に示すように、このディスクスナップショット部3は、ディスクスナップショット制御部5、揮発性メモリ6から構成される。この揮発性メモリ6内には、書き込みの時間的順序を維持するためのタイムスタンプ7、ディスク装置4に書き込むデータをログ構造化して保持する書き込みバッファ8、および書き込みバッファ8内の空き領域と保持されている書き込みデータの論理アドレスの情報とを保持するバッファ管理テーブル9が格納される。一方、ディスク装置4内には、通常のデータの他に、スナップショット情報（SS情報）10が格納される。ディスクスナップショット制御部5は、これらタイムスタンプ7、書き込みバッファ8、バッファ管理テーブル9およびスナップショット情報10を管理して、ディスク装置4への読み出し/書き込みおよびスナップショット情報の参照を制御する。

【0021】図3はこの揮発性メモリ6内の書き込みバッファ8とバッファ管理テーブル9との関係を示す。ディスクスナップショット制御部5は、ファイルシステム2からディスク装置4への書き込みを要求された書き込みデータを、すぐにディスク装置4に書き込むのではなく、ブロック単位に分割して書き込みバッファ8に順番に詰めて（ログ形式に）格納していく。このとき、その書き込みデータのファイルシステム2から見た論理アドレスを、バッファ管理テーブル9が格納したバッファ領域に対応するエントリに保存する。そして、そのエントリにデータが割り当てられたことを示すフラグ“F”を立てる。よって、このバッファ管理テーブル9を調べることで、ファイルシステム2から受け取った書き込みデータを格納すべき次のバッファ領域を決めることができる。図3では、バッファ領域B9まで書き込みデータが格納されており、B0、B1、～、B9の論理アドレスが、LA134、LA99、～、LA678であることを表わしている。

【0022】また、ディスク装置4は、それぞれブロックサイズの整数倍（ K ）であるストライプと呼ぶ決められた単位（そのディスク装置の1トラック長に近いサイズが好ましい）で書き込みを行なう。また、実際のディスク装置4を合わせた全記憶容量よりも少ない容量のディスク装置としてファイルシステム2には見せている（ファイルシステムが最初に記憶容量を問い合わせたときに、実際よりも少ない容量を返す）。よって、ファイルシステム2から論理的に読み書きできる記憶領域の他に余分な記憶領域が確保されることになる。ここでは、この領域を空領域と呼ぶことにする（図4参

照)。また、以下では、ファイルシステム2から見えるディスクスナップショット部3が提供する論理的なディスク領域を論理ディスク装置と呼ぶことにし、一方、ディスク装置4が実際に提供する物理的なディスク領域を物理ディスク装置と呼ぶことにする。

【0023】タイムスタンプ7は、ファイルシステム2からの書き込みデータが実際にディスク装置4に書き込まれるときに付加される情報であり、ディスク装置4内でのデータ書き込み順序を判定するのに用いられる。よって、書き込みバッファ8のデータがディスク装置4に書き込まれるごとにタイムスタンプ9がインクリメントされる。また、スナップショット情報10は、スナップショット採取時点のタイムスタンプの値を保持する。

【0024】以下、この実施形態のコンピュータシステムにおける各処理の動作原理を詳細に説明する。

【0025】まず、この実施形態のコンピュータシステムにおける書き込み処理について説明する。アプリケーションプログラム1a~1cがファイルシステム2上のファイルを書き込むと、ファイルシステム2は、対応する論理ディスク装置に対する書き込み要求に変換してディスクスナップショット部3に引き渡す。ディスクスナップショット制御部5は、ファイルシステム2から書き込むべきデータとその論理アドレスを受け取り、揮発性メモリ6上の書き込みバッファ8の空領域にブロック単位に分割して詰めて格納する。また、受け取った論理ディスク装置のアドレス（以下論理アドレスと呼ぶ）は、ブロックごとのアドレスに変換してバッファ管理テーブル9の対応するエントリに格納する。なお、すでに書き込みバッファ8に格納されているデータに対する更新データの場合には、書き込みバッファ8の空領域に詰めて格納するのではなく、直接書き込みバッファ8内の旧データを変更する。

【0026】そして、ディスクスナップショット制御部5は、ファイルシステム2から受け取った書き込みデータが1ストライプ分に1ブロック少ない数($K-1$)だけ書き込みバッファ8に溜まった段階で、それらのデータをディスク装置4に書き込みに行く。このとき、ディスクスナップショット制御部5は、最後の書き込みブロックとして、書き込み管理テーブル7に格納された各ブロックの論理アドレスと揮発性メモリ6上のタイムスタンプとから論理アドレスタグブロックを作成する（この論理アドレスタグブロック内のアドレスデータとデータブロックとは1対1の関係が予め設定されており、各データブロックの論理アドレスが分かるようになっている）。その後、ディスクスナップショット制御部5は、この論理アドレスタグブロックを加えた1ストライプ分のデータをまとめてディスク装置4の空領域に同時に書き込む（図5参照）。また、タイムスタンプ5の値は、書き込みが完了した段階でインクリメントされる。

【0027】このように書かれたデータブロックの読み

出し処理について説明する前に、まず無効ブロックの判定処理について説明する。

【0028】ここでは、図6に示す順番でファイルシステム2から1ブロックサイズのデータ書き込みがある場合を例に説明する。なお、図6中の L_n はファイルシステム2から渡される論理アドレスを表し、 S_n は書き込み順番を表わす。また、この実施形態のコンピュータシステムでは、書き込みバッファ8は15ブロック分のデータを保持できるものとする。この場合、最初の $S_1 \sim S_{15}$ の書き込みデータが1つのストライプ(ST_1)にまとめられ、タイムスタンプ TS_1 が付加されてディスク装置4の空領域に書き出される。同様に、 $S_{16} \sim S_{30}$ の書き込みデータが別のストライプ(ST_2)にまとめられ、タイムスタンプ TS_2 が付加されて別の空領域に書き出される。なお、書き出しごとにタイムスタンプはインクリメントされるので、 TS_1 と TS_2 とは $TS_1 < TS_2$ の関係をもつ。

【0029】この図6から分かるように、論理アドレス L_9 、 L_{18} のデータは、タイムスタンプ TS_1 のストライプでは S_5 、 S_2 のブロックとして、タイムスタンプ TS_2 のストライプでは S_{19} 、 S_{21} のブロックとして重複して存在する。書き込まれた順番を考えると、 S_{19} 、 S_{21} のデータブロックが有効であり、 S_5 、 S_2 のデータは無効と判定されなければならない。しかしながら、ここで便宜上使った書き込み順番 S_n は、実際のディスク装置4上には記録されていない。

【0030】そこで、この実施形態のコンピュータシステムでは、ストライプ内の論理アドレスタグを使ってこの判定を行なう。図6の例における2つのストライプ ST_1 、 ST_2 の論理アドレスタグ TG_1 、 TG_2 の内容は図7に示す通りである。図7から分かるように、2つの論理アドレスタグ TG_1 、 TG_2 に同じ論理アドレス L_9 、 L_{18} のデータが含まれており、ストライプ ST_1 のブロック B_5 、 B_2 と、ストライプ ST_2 のブロック B_4 、 B_6 とのどちらかのデータが無効である。さらに、論理アドレスタグ TG_1 のタイムスタンプ TS_1 と論理アドレスタグ TG_2 のタイムスタンプ TS_2 を比較すると、 $TS_1 < TS_2$ であることからストライプ ST_1 のブロック B_5 、 B_2 が無効であることが判定できる。以上のように、ディスク装置4内の論理アドレスタグを調べることにより、無効なデータブロックを見つけることができる。

【0031】次に、この実施形態のコンピュータシステムにおけるデータブロックの読み出し処理について説明する。原理的には、先に述べた無効ブロック判定の処理をディスク装置4内の全ストライプの論理アドレスタグに対して行ない、結果として残る読み出し要求のあった論理アドレスに対する有効ブロックの物理的位置を検出する。

【0032】しかしながら、読み出し要求がくる度にこ

の処理を行なっているのでは、ブロック読み出しに膨大な時間がかかってしまい実用的ではない。そこで、システム起動時にだけ全ストライプの論理アドレスタグの調査を行ない、揮発性メモリ6上に論理アドレスから物理アドレスへの変換マップを作る。ファイルシステム2からの読み出し要求に対しては、この変換マップを使って有効ブロックへのアクセスを行なう。これにより、常にアドレスタグの調査をしなくても良く、読み出し時に性能が大きく低下することはない。この変換マップについて図8を参照して説明する。図8に示すように、変換マップは各論理アドレスに対するブロックが格納されているストライプ番号ST#、そのストライプ内のブロック番号BLK#、およびそのタイムスタンプTS#をテーブル形式で保持している。よって、論理アドレスが与えられれば、このテーブルを引くことにより、ST#とBLK#とから簡単に実際の物理アドレスが求まる。

【0033】また、システム起動時の変換マップの作成は、調査した論理アドレスタグの全論理アドレスについて、テーブルのタイムスタンプより論理アドレスタグのタイムスタンプが大きいときだけ、そのストライプ番号と対応するブロック番号をテーブルに登録する。この調査を全ストライプについて行なえば、有効ブロックだけを指す変換マップができあがる。さらに、ディスク装置4にストライプを書き込むごとに、その論理アドレスタグに対して同様の処理を行なうことにより、この変換マップは常に有効なブロックを指す。このようにして、ディスク装置4上の物理ブロックの有効/無効の判定を行ない変換マップが作られる。

【0034】次に、この実施形態のコンピュータシステムにおけるスナップショットの採取処理について説明する。アプリケーションプログラム1a~1cやシステム管理者からのスナップショット採取の指示を受けると、ディスクスナップショット制御部5は書き込みバッファ8にその時点で溜まっているデータをディスク装置4に書き込みに行く。ただし、通常書き込みのように、書き込みバッファ8内に1ストライプ分に1ブロック少ない数(K-1)のデータブロックが溜まっているとは限らないので、まだ埋まっていない部分に対応する書き込み管理テーブル7には無効であることを示すヌルアドレスを格納する。そして、通常書き込みと同様に、最後の書き込みブロックとして、書き込み管理テーブル7に格納された各ブロックの論理アドレス(ヌルアドレスを含む)と揮発性メモリ6上のタイムスタンプとから論理アドレスタグブロックを作成する。その後、この論理アドレスタグブロックを加えた1ストライプ分のデータをまとめてディスク装置4の空領域に同時に書込む(図9参照)。さらに、このときに付加したタイムスタンプの値をスナップショット採取時の情報として、ディスク装置4内のスナップショット情報10に書き込む。そして、タイムスタンプ7の値は、書き込みが完了した段階

でインクリメントされる。このように、スナップショット採取には書き込みバッファ8に溜まっていたブロックとスナップショット情報10とをディスクに書き込むだけの時間しかかからない(その他の処理は非常に短い)ので、従来と同様にアプリケーションプログラム1a~1cを停止させることなくスナップショットが作成できる。

【0035】次に、図10を参照してこの実施形態のコンピュータシステムにおけるスナップショットの参照処理について説明する。まず、先にブロック読み出し処理で述べた無効ブロック判定の処理を、ディスク装置4内のタイムスタンプ値がスナップショット情報10の値よりも小さいストライプの論理アドレスタグに対して行ない、全論理アドレスに対する有効ブロックの物理的位置を検出する。読み出し処理では、全ストライプが対象だったが、このスナップショット参照処理では、スナップショット情報10のタイムスタンプ値までに書き込まれたストライプだけを対象にする点が異なる。つまり、図10のディスク状態では、ストライプST1~ST4がブロック読み出しの無効ブロック判定の対象となるが、スナップショット参照処理では、ストライプST1~ST3が無効ブロック判定の対象になる。

【0036】また、毎回無効ブロックの判定を行なっていたのでは、スナップショット参照に膨大な時間がかかってしまい実用的ではない。そこで、先の読み出し処理と同様に、最初にスナップショットを参照するときに対象となるストライプの論理アドレスタグの調査を行ない、揮発性メモリ6上に論理アドレスから物理アドレスへのスナップショット用変換マップを作る。図11に、ディスク状態が図10である場合のスナップショット用変換マップと読み出し用変換マップとの様子を示す(簡単化のためタイムスタンプのフィールドは省略した)。図11からわかるように、スナップショット採取後に更新された論理アドレスL3、L7、L9、~、L25、L27のブロックも、スナップショット用変換マップ(図11(a))からは更新前のブロックが参照できるようになっている。また、読み出し用変換マップ(図11(b))は常に最新のブロックを指している。このように、無効ブロック判定処理の対象とするストライプを変えることにより、スナップショット参照用と読み出し用の変換マップとが作成できる。

【0037】なお、スナップショット採取時のディスクイメージは、ディスクスナップショット部3経由で参照することができるが、その中のファイルを直接参照することはできない。そこで、図12に示すように、ディスクスナップショット部3にスナップショット採取時のディスクイメージを別の擬似ディスクとして見せる擬似ディスク機能をもたせる。一般にオペレーティングシステムは、起動フェーズで検出したディスクに格納されているデータがどのファイルシステムのものか検出し、対応

するファイルシステムを構築する機能を持っている。よって、この擬似ディスクを検出したオペレーティングシステムは、その上にスナップショット参照用ファイルシステム11を構築して、スナップショット参照用ユーティリティプログラム12がスナップショット採取時のファイルをアクセスできるようにする。このスナップショット参照用ユーティリティプログラム12としては、たとえば、特定ファイルを磁気テープにバックアップするプログラムなどが考えられる。なお、この擬似ディスク機能は、RAMディスクなどのハードウェアデバイスをもたないソフトウェアドライバとして実装すれば良いため、その実装は容易である。

【0038】(第2実施形態)次に、この発明の第2実施形態を説明する。

【0039】前述の第1実施形態のコンピュータシステムにおけるスナップショット採取処理では、ファイルシステム2が更新データをキャッシュする機能をもたないことを想定していた。つまり、アプリケーションプログラム1a~1cによるファイルの変更は、すぐにディスク装置4に書き出されるものと考えていた。よって、スナップショット採取時には、それまでに書き込みバッファ8に溜まっていた書き込みデータをディスク装置4に書き出すだけで、正しいファイルシステム2の状態がディスク装置4に保存できた。しかしながら、たとえばUNIXのファイルシステムなど、更新データをキャッシュする機能をもったファイルシステムも今日では広く使われている。この場合、先のスナップショット採取処理では、キャッシュされていた更新データが失われてしまうので、正しいスナップショットが採取できない。

【0040】そこで、この第2実施形態のコンピュータシステムでは、ファイルシステム2が更新データをキャッシュする機能を有する場合のスナップショット採取を説明する。

【0041】一般に、更新データをキャッシュするファイルシステムでは、アプリケーションプログラムにより行なわれたファイル変更が確実にディスク装置に記録され残ることを保証できるように、アプリケーションプログラムからの指示によりキャッシュしている更新データをすべて書き戻す機能をもっている。そこで、この第2実施形態のディスクスナップショット部3では、アプリケーションプログラム1a~1cからの書き出し前指示と書き出し後指示とを受け付け、これらの間にファイルシステム2からの書き出しが無い場合にだけ、先に述べたスナップショット採取処理を行なう条件付き採取機能を備える。

【0042】そして、このディスクスナップショット部3が備える機能を利用して、更新データがすべて書き戻された状態のスナップショットを、図13のフローチャートで示される手順を含んだアプリケーションプログラム1a~1cの指示に応じて採取する。アプリケーション

プログラム1a~1cは、まず、ディスクスナップショット部3に書き出し前指示を発行してこれからファイルシステム2の書き戻しを行なうことを指示する(ステップA1)。次に、キャッシュされている全更新データをディスク装置4に書き戻すようにファイルシステム2に指示する(ステップA2)。この指示を受けたファイルシステム2は、全更新データの書き戻しを実行し、その実行が完了したときに、その旨をアプリケーションプログラム1a~1cに通知する。一方、アプリケーションプログラム1a~1cでは、この書き戻しが完了した旨の通知をファイルシステム2から受け取ったときに、ディスクスナップショット部3に書き出し後指示を発行する(ステップA3)。

【0043】そして、この指示を受けたディスクスナップショット部3は、書き出し前指示から書き出し後指示までの間にファイルシステム2からの書き出しが無い場合にだけ、先に述べたスナップショット採取処理を行なう。ディスクスナップショット部3は、スナップショットを採取したか否かを書き出し後指示の応答としてアプリケーションプログラム1a~1cに返す。一方、アプリケーションプログラム1a~1cは、ディスクスナップショット部3がスナップショットを採取しなかった場合には(ステップA4のNO)、最初からこれまでの処理を繰り返し(ステップA1~ステップA3)、採取した場合に(ステップA4のYES)、この処理を終了する。

【0044】以上の処理により、ファイルシステム2へ書き戻しを指示してもディスク装置4へ書き出すべきデータが無かったこと、すなわち、ディスク装置4上にはすでに正しいファイルシステム2の状態が保存されていることが保証される。したがって、ディスクスナップショット部3で正しいスナップショットが採取できることになる。たとえば、ただ単にファイルシステム2に書き戻し指示をするとともにディスクスナップショット部3にスナップショット採取を指示したのでは、この2つの指示の間にファイル更新が行なわれた場合に、一部の更新データがキャッシュに残ってしまうため、正しいスナップショットが採取できない危険性がある。よって、ここで述べた手順が必要になる。

【0045】(第3実施形態)次に、この発明の第3実施形態を説明する。

【0046】前述の第1実施形態のコンピュータシステムでは、スナップショットは1つしか採取できなかったが、この第3実施形態のコンピュータシステムでは、複数のスナップショットを採取する。

【0047】まず、先の第1実施形態と異なるのは、図14に示すように、ディスク装置4上のスナップショット情報10が、複数のスナップショットそれぞれのタイムスタンプを保持する構造となっていることである。

【0048】そして、スナップショット採取処理におい

ては、その時点で溜まっていたデータを書き込みにいく点は同じであるが、書き込んだストライプのタイムスタンプ値を、そのスナップショット(SS)のスナップショット情報10としてディスク装置4に保存する。たとえば、図14のストライプ3(ST3)をスナップショットの採取処理で書き込む場合には、そのタイムスタンプ値TS3をスナップショット1(SS1)のスナップショット情報10として保存する。同様に、ストライプ5(ST5)のスナップショット採取処理では、そのタイムスタンプ値TS5をスナップショット2(SS2)のスナップショット情報10として保存し、ストライプ8(ST8)のスナップショット採取処理では、そのタイムスタンプ値TS8をスナップショット3(SS3)のスナップショット情報10として保存する。

【0049】また、スナップショット参照処理においては、図14に示すように、参照するスナップショットによって対象となるストライプが異なるだけで基本的処理は同じである。たとえば、スナップショット2(SS2)を参照する場合には、スナップショット情報10のタイムスタンプ値TS5から、ST1~ST5が対象となるストライプであることが分かる。そこで、ST1~ST5のストライプだけを調べて、無効ブロックの判定処理とスナップショット参照用変換マップの作成処理とを実行する。そして、SS2のスナップショットを参照する場合には、この変換マップを参照してスナップショット採取時のデータをアクセスする。同様に、SS1、SS3のスナップショットを参照する場合には、そのタイムスタンプ値TS3、TS8から対象となるストライプがST1~ST3、ST1~ST8であることが分かり、それらの対象となるストライプに対してだけ無効ブロックの判定処理とスナップショット参照用変換マップの作成処理とを実行し、その変換マップを使ってそれぞれのスナップショットが採取されたときのデータをアクセスする。

【0050】このように、スナップショット情報10として複数のタイムスタンプ値を保存できるようにするだけで、複数のスナップショットを取り扱えるようになる。

【0051】次に、ディスクスナップショット部3が採取した複数のスナップショットをスナップショット参照用ユーティリティプログラム12からアクセスする方法を説明する。

【0052】第1の方法としては、図15に示すように、ディスクスナップショット部3を拡張し、採取したスナップショットごとに対応する擬似ディスク装置13a~13cを見せる機能をもたせて、それぞれの擬似ディスク装置13a~13cの上にスナップショット参照用ファイルシステム11a~11cを構築する。そして、それぞれのスナップショットをアクセスするスナップショット参照用ユーティリティプログラム12は、対

応するスナップショット参照用ファイルシステム11a~11cを経由して、スナップショット採取時のアクセスを行なう。しかしながら、この方法では、スナップショット数が増えると擬似ディスクも増えてしまう。たとえばマイクロソフト社のWindowsなどのように、プログラムから見えるディスク台数に上限があるオペレーティングシステムも多い。したがって、擬似ディスク装置13a~13cが増えてしまうこの方法では、この制限のために適用できない場合も多くなる。また、プログラムから見えるディスク台数が多いとコンピュータシステムそのものの管理も大変である。

【0053】そこで、第2の方法としては、図16に示すように、ディスクスナップショット部3にスナップショット用として擬似リムーバブルディスク13を見せる機能をもたせて、各スナップショットはその擬似リムーバブルディスク用の擬似メディアとして取り扱う。この方法では、擬似ディスク装置13が1台しか要らないので、第1の方法のような問題は発生しない。なお、この方法では、複数のスナップショットを同時にアクセスできないが、同時にスナップショットへのアクセスが必要なケースは少ないので、大きな問題にはならないと思われる。また、それぞれのスナップショットの必要なファイルを最新のディスクイメージにコピーすれば、同時アクセスが可能となるので大きな制限ではない。

【0054】また、通常、リムーバブルディスクをサポートするオペレーティングシステムでは、メディアを変更するとそれに合わせてファイルシステムの再構築などが自動的に行なわれる。そこで、スナップショットをSS1からSS2へ切り替えるときには、まず、この擬似リムーバブルディスク13からメディアが変更された旨の情報をスナップショット参照用ファイルシステム11やオペレーティングシステムに通知する。すると、オペレーティングシステムは、変更されたメディアをアクセスするために、これまでアクセスしていたスナップショットSS1に対するスナップショット参照用ファイルシステム11を終了し、変更されたメディアに対するスナップショット参照用ファイルシステム11を起動する。この新しく起動されたスナップショット参照用ファイルシステム11からのアクセスに対して、ディスクスナップショット部3の擬似リムーバブルディスクは、スナップショットSS2のディスクイメージを返すことにより、スナップショット参照用ユーティリティプログラム12は、スナップショットSS2のファイルをアクセスできるようになる。なお、ディスク(擬似ディスクを含む)がリムーバブルか否かの情報は、システム起動時などにオペレーティングシステムが問い合わせるので、このときに、リムーバブルである旨をディスクスナップショット部3がオペレーティングシステムに伝えればよい。

【0055】以上をまとめると、スナップショットを格

納するディスクを擬似リムーバブルディスクとして扱うためにディスクスナップショット部3が行なうことは、
(1) システム起動時などでのオペレーティングシステムからの問い合わせにリムーバブルである旨の情報を返す。

【0056】(2) ユーザやアプリケーションプログラムからのスナップショット切り替え指示を受け付ける。

【0057】(3) このスナップショット切り替え指示に対して、メディアが変更された旨の情報をファイルシステムやオペレーティングシステムに通知する。

【0058】(4) オペレーティングシステムやファイルシステムからの変更されたメディアへのアクセスに対しては、ユーザやアプリケーションプログラムから指定されたスナップショットのイメージをアクセスできるようにする。

【0059】となる。

【0060】しかしながら、この第1および第2の方法では、オペレーティングシステムやスナップショット参照ユーティリティプログラム12を含む各種プログラムから見える、スナップショットを採取したときのディスク装置とスナップショットを参照する場合のディスク装置とが異なってしまう。たとえば、図15の場合、Windowsなどのオペレーティングシステムからは、ディスク装置4は“C:”ドライブ、SS1の擬似ディスクは“D:”ドライブなどとして見えてしまう。よって、環境変数などで各種プログラムが使うべきファイル名として、“C:”ドライブのAファイルとあったとしても、スナップショット参照時には、“D:”ドライブのAファイルとしなければ、各種プログラムは正しく動作しない。同様に、スナップショットのファイルの中にある処理に関連するファイル名として“C:”ドライブのBファイルとあったとしても、スナップショットの参照時には、“D:”ドライブのBファイルとしなければ、各種プログラムは正しく動作しない。

【0061】以上のように、オペレーティングシステムや各種プログラムから見えるディスク装置が異なってしまうと、環境変数やファイル内容をそれに合わせて変更しないとプログラムが正しく動作しない場合も多い。一般に、どのファイルや環境変数を変更すればよいか判っておらず、スナップショットを参照する度に変更するのは大変な作業である。

【0062】そこで、第3の方法として、図17に示すように、ディスクスナップショット部3にディスク装置4がリムーバブル機能をもつようにオペレーティングシステム、ファイルシステムおよび各種プログラムに見せるリムーバブル化機能を設け、最新のディスクイメージと複数のスナップショットとをそれぞれ別の擬似メディアとして取り扱う。この方法では、スナップショット採取時の最新ディスクイメージとスナップショットのイメージとは擬似メディアが異なるだけであるので、オペレ

ーティングシステムや各種プログラムなどから見えるディスク装置は同じであり、第1および第2の方法のような問題は発生しない。なお、この方法では、最新のディスクイメージとスナップショットとを同時にアクセスできなくなるが、ディスクやファイル名を意識するようなプログラムはもともと複数同時には実行できないので問題は無いと思われる。

【0063】先に述べたように、ディスク装置(擬似ディスクを含む)がリムーバブルか否かの情報は、システム起動時などにオペレーティングシステムが問い合わせてくるので、ディスク装置4をリムーバブル化するには、このオペレーティングシステムからディスク装置4への問い合わせを途中で獲得して、リムーバブルである旨の情報を返せばよい。また、最新ディスクイメージやスナップショットを切替えるときには、第2の方法で述べたメディア変更の手順を行えばよい。

【0064】ここで、システムディスクのスナップショット参照について考える。システムディスクのスナップショットイメージでオペレーティングシステムを起動できれば、スナップショットを採取した時点の過去のオペレーティングシステム状態をそのまま再現でき、トラブル対応などで非常に役に立つ。しかし、システムディスクとは、オペレーティングシステムが実行するのに必要なオペレーティングシステムコード、デバイスドライバ、スワップファイルなどを格納するディスクであって、システム実行中は常にアクセスできなければならない。よって、システム実行中にアクセスできなくなるリムーバブルディスクにはできず、第2および第3の方法を適用することはできない。また、システムディスクは“C:”ドライブなど決まっている場合も多く、オペレーティングシステムから見えるディスクが異なってしまう第1および2の方法も適用できない。

【0065】そこで、第4の方法として、図18に示すように、システム起動時に選択されたスナップショットを最新のディスク装置4のイメージの代わりにオペレーティングシステムやファイルシステムに見せるイメージ選択機能をディスクスナップショット部3に設け、任意のスナップショットをシステム起動時に選択できるようにする。選択したディスクイメージは、システム実行中には替わらないのでアクセスできなくなることはない。また、オペレーティングシステムやアプリケーションプログラムから見えるドライブは、スナップショットを採取したときと同じであるので、システムディスクが“C:”ドライブなどに予め決まっても問題がない。

【0066】図19に、この方法におけるコンピュータシステムの運用例を示す。通常のアプリケーションプログラムを実行する場合は、システム起動時にディスクスナップショット部3のイメージ選択機能により最新のディスクイメージをユーザなどが選択して起動処理を行な

う。これにより、最新のディスクイメージに対応した最新のファイルシステムが構築され、通常のアプリケーションを実行できるようになる(T1)。ここで、スナップショットSS1用のアプリケーションを実行するには、一旦、現在実行中のシステムを終了してシステムを再起動する。そして、この再起動時に、ディスクスナップショット部3のイメージ選択機能によりスナップショットSS1をディスクイメージとしてユーザなどが選択する。これにより、スナップショットSS1に対応したSS1用のファイルシステムが構築され、スナップショットSS1のファイルにアクセスするアプリケーションプログラムを実行できるようになる(T2)。さらに、通常のアプリケーションプログラムを実行するには、システムを再起動して、再起動時に最新のディスクイメージを選択するようにする。

【0067】なお、図19ではユーザなどがイメージを選択するのはシステム起動時だけであったが、ディスクスナップショット部3に次のシステム起動時に選択すべきイメージを記録する次回記録機能を持たせて、イメージ選択機能はこの次回記録機能に記録されたイメージを選択するようにもできる。これにより、システム起動時以外でも、ユーザなどだけでなくアプリケーション(たとえばT1期間に実行されるアプリケーションプログラム)からでもイメージを選択できるようになる。この次回記録機能は、ディスク装置4の予め決めた場所に選択すべきイメージを保存しておき、システム起動時にこの情報を読み出してイメージ選択機能に教えるだけの簡単なプログラムで実現できる。

【0068】最後に、アプリケーションプログラムからスナップショットにアクセスするこれら第1乃至第4の方法は、それぞれ排他的に適用されるのではなく、任意に組み合わせて適用することも可能である。

【0069】ところで、これまでスナップショットの識別情報としては、SS1~SS3を使っていたが、これだけではいつの時点のスナップショットか分かりにくく、一旦スナップショットを参照してから別のスナップショットを参照すべきことが判明する場合も多い。そこで、図14に示すように、スナップショット情報10としてスナップショットを採取した日時を保存するフィールドを用意し、ディスクスナップショット部3がスナップショット採取時のタイムスタンプだけでなく、日時もスナップショット情報に記録するようにする。そして、この日時をスナップショットに対応した擬似ディスクや擬似メディアの識別情報として用いることにより、いつの時点のスナップショットかすぐに分かり、効率のよいスナップショットの運用が可能になる。

【0070】(第4実施形態)次に、この発明の第4実施形態を説明する。

【0071】これまでに述べた実施形態では、一度採取したスナップショットは参照だけで変更しないことを前

提に説明してきた。よって、図20(a)に示すように、各スナップショットは採取された時間順に一直線に並んでおり一番先頭が最新のディスクイメージとなる順序関係があった。しかしながら、先に述べたシステムディスクのスナップショットの場合、オペレーティングシステム起動途中やオペレーティングシステム実行中にファイルの作成、変更、削除などが行なわれ、また、グループウェアやDBMSなどのアプリケーションプログラムからスナップショットのファイルにアクセスする場合も、アプリケーションプログラムが実行中である旨の情報などがファイルに書き込まれる。しかも、古いスナップショットの状態から起動したシステムを使って次にスナップショットを採取しても、それは古いスナップショットに戻る前の最新イメージKとも異なる。このような場合、戻る前の最新イメージKも参照したくなる場合があると考えられる。つまり、図20(b)に示すように、スナップショットSS1、SS2、SS3の順番でスナップショットを採取した後で、SS2のスナップショット時点から再起動したシステムでは、SS3への変更とは全く異なる変更を行ない、さらにそれに対するスナップショットSS4を採取するなどができなければならない。また、図20(b)に示すように、複数の最新ディスクイメージを並行してアクセスできる必要もある。

【0072】以下ではスナップショットの変更方法について説明する。

【0073】まず、スナップショットの変更を可能にするには、図21に示すように、ディスク装置4上のスナップショット情報として、親となるスナップショット情報(親SS)と、親SSからそのスナップショットまで書き出されたストライプのタイムスタンプを保存するようにする。たとえば、図14のスナップショットSS3は、その基となったスナップショットはSS2であるので、スナップショット情報の親SSとしてSS2を保存する。また、SS2からSS3への間に書き出されたストライプのタイムスタンプは、TS6~TS8であるので、スナップショット情報のタイムスタンプとしてTS6~TS8を保存する。また、複数の最新ディスクイメージを実現するため、スナップショット情報と同様に、親SSとその親SSからその最新イメージまで書き出されたストライプのタイムスタンプとを最新イメージ(NI)情報としてディスク装置4上に保存する。

【0074】ここで、図20(a)の状態から図20(b)のように、スナップショットSS2を変更してスナップショットSS4を作成する場合を考える。まず、先に述べた擬似ディスクなどを使ってSS2の参照を行なう。そして、オペレーティングシステムやアプリケーションプログラムがこのスナップショットSS2を更新し、それに対する書き込みブロックがディスクスナップショット部3に到着した段階で、ディスクスナップショ

ット部3はSS3の先にある最新イメージNI1への追加的更新でないので、最新イメージ情報として最新イメージNI2を作成する。(親SSはSS2であるので、この情報がNI2の親SSとして保存される。)さらに、通常のディスクスナップショット部3の書き込み動作と同じように、メモリ6上の書き込みバッファ8にこの書き込みブロックを順次溜め、書き込みデータが1ストライプ分に1ブロック少ない数(K-1)だけ書き込みバッファ8に溜まった段階で、ディスク装置4に書き込みに行く。最後の書き込みブロックとして、各ブロックの論理アドレスとタイムスタンプとから論理アドレスタブロックを作成するのも同じである。なお、通常の最新イメージNI1とSS2の更新とを同時に行なう場合には、スナップショット変更用に別の書き込みバッファ8を用意して、それぞれ別々に溜める。そして、ディスク装置4に書き込んだストライプST10のタイムスタンプTS101を最新イメージNI2のタイムスタンプとして記録する(図21参照)。

【0075】SS2の変更が進んでスナップショットSS4を採取する処理も前述のスナップショット採取処理と同じである。つまり、まだ埋まっていない部分に対応する書き込み管理テーブル9にヌルアドレスを格納して、書き込みバッファ8にその時点で溜まっているデータをディスク装置4に書き込みに行く。そして、最新イメージNI2の情報(親SSがSS2、タイムスタンプがTS101)とこのときに付加したタイムスタンプの値TS102とから、スナップショットSS4のスナップショット情報として、親SSとしてSS2、タイムスタンプとしてTS101~TS102をディスク装置4へ書き込む。そして、タイムスタンプの値は書き込みが完了した段階でインクリメントされる。また、最新イメージNI2の最新イメージ情報は、親SSとして新しく作成したSS4が保存され、タイムスタンプはクリアされる。

【0076】さらに、このスナップショットSS4が更新されストライプST13がディスク装置に書き出された場合は、最新イメージNI2のタイムスタンプ情報にはTS103が保存される。

【0077】次に、スナップショットを変更する場合の最新イメージやスナップショットの参照処理について説明する。たとえば、ディスク装置4が図21の状態であるときに、スナップショットSS4を参照する場合を考える。このSS4は、SS2を元に変更を加えたスナップショットであるので、参照時の変換マップ作成の対象とすべきストライプはSS2対象のST1~ST5とSS2からSS4への変更分のST10~ST11とでなければならない。そこで、これらのストライプを以下の手順で見つける。

【0078】まず、スナップショット情報を使って、SS4の親SSがSS2であることとSS2からSS4へ

の更新データを書き出したストライプのタイムスタンプがTS101~TS102であることを知る。次に、スナップショット情報10を使って、SS2の親SSがSS1であることとSS1からSS2への更新データを書き出したストライプのタイムスタンプがTS4~TS5であることを知る。この処理を親SSが無くなるまで繰り返すことにより、スナップショットSS4までに書き出されたストライプのタイムスタンプが、TS1~TS3、TS4~TS5、TS101~TS102であることが分かる。

【0079】よって、ディスク上の各ストライプのタイムスタンプを比較することにより、SS4参照時の対象ストライプであるST1~ST5とST10~ST11とを見つげられる。なお、先に説明した実施形態では、最新ディスクイメージを読み出す場合には、ディスク装置上の全ストライプを対象に変換マップを作成すれば良かったが、ストライプの変更が可能な場合には、最新イメージ情報とスナップショット情報とを使い、スナップショット参照と同様の処理を行なって対象とすべきストライプのタイムスタンプを求める。

【0080】なお、前述のスナップショット変更処理の説明では、ストライプを書き込む度に最新イメージ(NI)情報のタイムスタンプ情報を変更していた。また、スナップショット情報や最新イメージ情報のタイムスタンプ情報などは、図21に示すように、区間情報の方が扱い易い。そこで、最新イメージ(NI)を作成するときに、書き出す際に用いるタイムスタンプ値を予め予約しておき、ディスク装置にタイムスタンプ(TS)情報として保存しておく方法が考えられる(図21参照)。これにより、ストライプを書き出す度に最新イメージ情報のタイムスタンプを更新しなくても、ディスク上の全ストライプのタイムスタンプを調べることによって、この情報を再現できる。たとえば、ストライプST13を書き出したときに最新イメージNI2のタイムスタンプを更新しなくても、最新イメージNI2のタイムスタンプはTS101~TS200が割り当てられることと、ディスク装置4上のストライプでこの区間に入る最大のタイムスタンプはTS103であることから、NI2のタイムスタンプは最後がTS103であることが分かる。また、予めタイムスタンプ値を最新イメージごとに分けているので、スナップショット情報やNI情報のタイムスタンプ値は必ず区間情報になる。

【0081】ここで、図22を参照してスナップショットの削除について考える。スナップショットの変更を許さない場合、図22(a)のようにスナップショットSS2を削除するにはスナップショット情報のスナップショットSS2に関する情報を削除するだけで良く、特別な処理を必要としない。しかしながら、スナップショットの更新が可能な場合には、スナップショットの削除処理はもっと複雑になる。まず、スナップショット情報や

最新イメージ情報には親SSを含む。よって、親SSが削除された場合には、削除された親SSの親SSをスナップショット情報や最新イメージ情報の親SSにする必要がある。また、スナップショット情報や最新イメージ情報のタイムスタンプは、削除された親SSのタイムスタンプも含むように修正する必要がある。

【0082】さらに、図21のディスク状態で図22(b)のようにスナップショットSS3と最新イメージNI1とを削除する場合には、図23に示すように、スナップショット情報とNI情報とを削除するだけでなく、削除ストライプ情報(DEL情報)としてスナップショットSS3と最新イメージNI1のタイムスタンプとをディスク装置4上に保存して、これらのタイムスタンプを使わないようにする必要がある。

【0083】(第5実施形態)次に、この発明の第5実施形態を説明する。

【0084】これまで述べた実施形態におけるスナップショット採取処理では、その時点で溜まっていた更新データだけで1つのストライプを構成しディスク装置を書き出していたので、スナップショット採取時のストライプには更新データが無く無駄である領域が出てしまう。したがって、スナップショット採取の頻度が多い場合には、ディスク装置4上に無駄な領域が増えてしまう。

【0085】そこで、スナップショット採取処理は先の場合と同じで、まだ埋まっていない部分に対応する書き込み管理テーブル9にヌルアドレスを格納し、書き込みバッファにその時点で溜まっているデータにタイムスタンプを付けてディスク装置4に書き込みに行く。このとき、スナップショット情報としては、図24に示すように、タイムスタンプ値だけでなく、スナップショット採取時に溜まっていたストライプ内のブロック番号も保存する。

【0086】そして、スナップショット採取直後の書き込み処理において、書き込みバッファ8やバッファ管理テーブル9をクリアすることなく、更新データを続けて書き込みバッファ8に格納する。そして、この書き込みバッファ8が1ストライプ分に1ブロック少ない数(K-1)だけ溜まった段階で、論理アドレスタグブロックを作成し、ディスク装置4に書き出す。このとき、論理アドレスタグブロックのタイムスタンプには、スナップショット採取時の値を、書き出すディスク装置4の位置は、スナップショット採取と同じストライプに上書きする。(図24のST3)このように採取したスナップショットの参照処理では、スナップショット情報のタイムスタンプ値とタイムスタンプが一致するストライプについては、スナップショット情報のブロックまでをそのスナップショットの対象として変換マップ作成時の無効ブロックの判定を行なう。つまり、図24の場合には、ストライプST3についてはブロックB1~B6がスナップショットの対象となるブロックとする。

【0087】スナップショット採取直後の書き出し処理とスナップショット参照処理とにおいて、以上のように処理することにより、スナップショットのストライプにも無駄な領域を作らないでスナップショット採取することができる。

【0088】(第6実施形態)次に、この発明の第6実施形態を説明する。

【0089】これまで述べた実施形態においてディスクスナップショット部3として採用している、更新前データの領域を書き換えるのではなく、更新データを溜めておいてディスク装置内の予め用意された別の空領域にまとめて書き込む方法では、空領域が常に存在することが必須である。そのため、ファイルシステム2からのディスクアクセスが空いている間に、すでに他の領域にデータが書き込まれて無効になっているデータを寄せ集めて空領域を作る必要がある。ここでは、この処理を詰替え処理と呼ぶものとする。この詰替え処理は、先に述べた無効ブロック判定とストライプ統合との2つのステップからなる。ここで、無効ブロックとは、スナップショット参照としても、通常の読み出しとしてもアクセスされないブロックのことである。つまり、スナップショット参照用変換マップおよび読み出し用変換マップのいずれからも指されていないブロックが無効ブロックである。

【0090】ストライプ統合の例として、図25に示すように2つのストライプST1、ST3を1つのストライプST10に統合する場合を考える。図25に示すように、ストライプST1では、B2、B7、B8、B12、B13の5ブロックが有効であり、他の10ブロックは無効であるとする。同様に、ストライプST3では、ブロックB18、B19、B20、B21、B22、B24、B25、B27、B29の9ブロックが有効であり、他の6ブロックが無効であるとする。この2つのストライプの有効ブロックは、合わせて14ブロックしかないので、この2つのブロックを1つに統合することにより、結果として1つの空領域が作れることになる。

【0091】ストライプ統合では、2つのストライプをメモリ内に読み出し、有効ブロックだけを詰めて書き込みバッファに移す。それに合わせて、論理アドレスタグも、図26に示すように、TG1、TG3から有効ブロックの論理アドレスだけを対応する位置に移して新しい論理アドレスタグTG10を作り、その時点のタイムスタンプを更新する。この例では、14個の有効ブロックしかなかったため、最後のデータブロックは空状態のまま書き込む。このときは、論理アドレスタグTG10の最後のデータブロックに対する論理アドレスにヌルアドレスを入れることによりデータが入っていないことを表わせるので問題はない。

【0092】しかしながら、スナップショットがある場合、ただ詰替えを行なったのでは詰替える前の領域を空

き状態にできない。図27を参照してこれらについて説明する。

【0093】たとえば、図27に示すように、ストライプST1とST3を詰替えて新しいストライプST10を作ったとしても、スナップショット情報では、スナップショットSS1はタイムスタンプがTS1からTS4であるストライプが対象であり、スナップショットSS1の参照では、空き状態であるはずのストライプST1およびST3のブロックをアクセスにいつてしまう。

【0094】そこで、図27に示すように、詰替え情報として、詰替えて作成されたストライプのタイムスタンプをディスク上に保存する。また、ストライプの参照処理では、スナップショット情報のタイムスタンプ値から決まるストライプだけでなく、詰替え情報のタイムスタンプのストライプも対象とする。たとえば、スナップショットSS1の参照処理では、タイムスタンプがTS4より小さいストライプ(ST1~ST4)だけでなく、詰替え情報のタイムスタンプのストライプ(ST10)も無効ブロック判定の対象とする。これにより、ストライプST10はストライプST1およびST3の有効ブロックだけを詰替えたものであることから、ストライプST1およびST3の全ブロックは必ず無効と判定され、ストライプST1およびST3はスナップショットからも参照されることはなくなる。よって、実質的には空き領域となる。

【0095】ただし、この方法は、同一のスナップショットのみから対象とされるストライプ同士を詰替えた場合にのみ正しく動作し、図28に示すような、スナップショットSS1、SS2、SS3が対象とするストライプST2と、スナップショットSS2、SS3が対象とするストライプST6とを詰替えて、ストライプST10を作るような場合には正しく動作しない。なぜならば、ストライプST10にはスナップショットSS1採取後に書き出されたストライプST6の更新ブロックが含まれているため、スナップショットSS1の参照処理でストライプST10も対象にしてしまうと、SS1採取時のディスク状態が再現できないからである。

【0096】そこで、図28に示すように、詰替えによって作成されたストライプの中に、元になった各ストライプのタイムスタンプと元ブロックが格納されているブロック位置とを元ストライプ情報として保存する。そして、スナップショットの参照処理において、詰替えによって作成されたストライプ(詰替え情報から分かる)に対しては、この元ストライプ情報も活用して不要なブロックを対象にしないようにする。つまり、先のスナップショットSS1の参照処理においては、ストライプST10の全ブロックを対象にするのではなく、スナップショットSS1のタイムスタンプTS4より小さい、ブロックB1~B6(タイムスタンプTS2)だけを対象にする。これにより、スナップショットSS1採取時のデ

ィスクイメージを再現できるとともに、ストライプST2の全ブロックが無効となりST2を空き領域にできる。

【0097】最後に、これまでの詰替え処理の説明では、無効ブロックとはスナップショット参照としても通常の最新イメージの読み出しとしてもアクセスされないブロックと定義し、スナップショット参照用変換マップと読み出し用変換マップとが作成され、それらの変換マップから参照されないブロックを見つける方法を取っていた。しかしながら、この方法では、スナップショット数が多くなると作成すべき変換マップの数も多くなり、非常に効率が悪い。

【0098】そこで、無効ブロックの判定に無効ブロックに対応したビットマップを使う方法について説明する。図29には無効ブロックに対応したビットマップを使った無効ブロック判定のための構成が示されている。

【0099】図29に示すように、メモリ6上には現時点の最新ディスクイメージを構成するディスク装置4上の全ブロックの状態をビットマップとして保持する。

【0100】このビットマップは、全ビットを無効状態に初期化した後、読み出し用変換マップから参照されるブロックに対応するビットを有効状態にすることにより容易に作成できる。また、書き出し処理により書き込みバッファ8内の更新データが空き状態のストライプに書き出されると、書き出したストライプのビットマップを有効状態にする(なお、書き込みバッファ管理テーブル9にヌルアドレスをセットされたブロックに対応するビットは無効のままとする)。さらに、この書き出しにより変更された論理アドレスに対する書き出し以前のブロック位置を読み出し用変換マップから求めて、このブロックを無効化されたブロックとして対応するビットマップを無効状態にする。以上のビットマップ変更を更新データの書き出し処理ごとに行なえば、常に最新ディスクイメージを構成するディスク装置4上の全ブロックの状態を維持できるようになる。

【0101】そこで、チェックポイント採取処理では、図29に示すように、このビットマップもスナップショット情報として保存する。そして、無効ブロックの判定処理では、メモリ6上のビットマップとスナップショット情報の全ビットマップとを調べることにより、無効状態にあるブロックを無効ブロックと判定する。これにより、スナップショットの変換マップを作成しなくても、無効ブロックの判定が可能となり、詰替えるべきストライプを容易に見つけることが可能となる。

【0102】なお、図29では、メモリ6の全ビットマップをスナップショット情報として保存したが、このビットマップは無効ブロックの判定に用いるだけなので、図30に示すように、スナップショット採取時に無効ブロックの多いストライプに対するビットマップだけをスナップショット情報として保存することもできる。図3

0の例では、ストライプST3、ST5、ST6、ST10の無効ブロックが多いので、これらのストライプのビットマップだけがストライプ情報として保存されている。これにより、スナップショット情報のための領域を減らせ、詰替えのための無効ブロック判定も効率よく行なえる。

【0103】(第7実施形態)次に、この発明の第7実施形態を説明する。

【0104】これまでに述べた実施形態では、更新データやスナップショット情報、最新イメージ情報、削除ストライプ情報、詰替え情報を記憶するディスク装置は1台であるものとして説明してきたが、図31に示すように、複数のディスク装置から1台の論理的ディスク装置を構成したディスクアレイ装置(RAID0、1、3、4、5)であっても良い。このとき、これまでの説明で使ってきたストライプのサイズとしては、RAID0の場合は、“ストライプユニット単位(そのディスク装置の1トラック長に近いサイズが好ましい)×ディスク台数”を選び、一方、RAID3、4、5の場合は、“ストライプユニット単位×(ディスク台数-1)”を選ぶことにより、全ディスクを同じタイミングで書き込めるため非常に効率が良い。特に、冗長情報としてパリティを使うRAID4、5では、通常必要な2回の読み出しと1回の書き込みが不要になり、非常に効率が良い。

【0105】なお、この発明は、不揮発性記憶装置としてディスク装置を用いるコンピュータシステムだけではなく、光磁気ディスク、DVD-RAMおよびEEPROMなどのあらゆる不揮発性記憶装置を用いるスナップショットやバックアップを必要とするコンピュータシステムに適用可能である。

【0106】

【発明の効果】以上詳述したように、この発明では、たとえばディスク装置などの不揮発性記憶装置とファイルシステムとの間に、ディスクレベルのスナップショットを採取するスナップショット管理手段を介在させる構造を採用するため、既存のコンピュータシステムのファイルシステムをそのまま使え、専用のファイルシステムを新しく開発する必要がない。また、ファイルシステム自身もブロック木構造である必要はまったくなく、NTFSなどのエクステンツベースのファイルシステムにでも適用できる。さらに、従来発生していた、スナップショット作成後の更新性能が大きく低下するという問題もない。そして、スナップショットはスナップショット情報という簡単なデータ構造で実現されるため、スナップショットに対する操作は読み出しだけでなく更新も可能となり、ファイルをコピーしなくてもファイル更新を伴うアプリケーションプログラム(DBMS、メールなど)を使って過去のスナップショットをアクセスして必要な情報を取り出せる。また、スナップショットは擬似ディスク、リムーバブルメディア、最新ディスクイメー

ジとして見せることにより、スナップショットをアクセスするために、既存のオペレーティングシステム、ファイルシステムおよびアプリケーションプログラムなどを改造することなく、既存のコンピュータシステムにそのまま適用することが可能である。

【図面の簡単な説明】

【図1】この発明の第1実施形態に係るコンピュータシステムの構成を示す概念図。

【図2】同第1実施形態のディスクスナップショット部の構成を示す概念図。

【図3】同第1実施形態の揮発性メモリ内の書き込みバッファとバッファ管理テーブルとの関係を示す図。

【図4】同第1実施形態のディスク装置上の空領域を示す図。

【図5】同第1実施形態のコンピュータシステムにおける書き込み処理を説明するための図。

【図6】同第1実施形態の無効ブロックの判定処理を説明するための図。

【図7】図6の例における2つのストライプST1、ST2の論理アドレスタグTG1、TG2の内容を示す図。

【図8】同第1実施形態の論理アドレスから物理アドレスへの変換マップを示す図。

【図9】同第1実施形態のコンピュータシステムにおけるスナップショットの採取処理を説明するための図。

【図10】同第1実施形態のコンピュータシステムにおけるスナップショットの参照処理を説明するための図。

【図11】ディスク状態が図10である場合のスナップショット用変換マップと読み出し用変換マップとの様子を示す図。

【図12】ディスクスナップショット部がもつスナップショット採取時のディスクイメージを別の擬似ディスクとして見せる擬似ディスク機能を説明するための図。

【図13】この発明の第2実施形態に係るコンピュータシステムのスナップショット採取手順を説明するためのフローチャート。

【図14】この発明の第3実施形態に係るコンピュータシステムのスナップショット情報を示す図。

【図15】同第3実施形態の複数のスナップショットをアプリケーションプログラムからアクセスする第1の方法を説明するための図。

【図16】同第3実施形態の複数のスナップショットをアプリケーションプログラムからアクセスする第2の方法を説明するための図。

【図17】同第3実施形態の複数のスナップショットをアプリケーションプログラムからアクセスする第3の方法を説明するための図。

【図18】同第3実施形態の複数のスナップショットをアプリケーションプログラムからアクセスする第4の方法を説明するための図。

【図19】同第3実施形態の複数のスナップショットをアプリケーションプログラムからアクセスする第4の方法におけるコンピュータシステムの運用例を示す図。

【図20】この発明の第4実施形態に係るコンピュータシステムにおけるスナップショットの変更を説明するための図。

【図21】同第4実施形態のスナップショット情報、最新イメージ情報およびタイムスタンプ情報を説明するための図。

【図22】同第4実施形態のコンピュータシステムにおけるスナップショットの削除を説明するための図。

【図23】同第4実施形態のスナップショット情報、最新イメージ情報およびタイムスタンプ情報のスナップショットの削除に伴う取り扱いを説明するための図。

【図24】この発明の第5実施形態に係るコンピュータシステムのスナップショット情報を示す図。

【図25】この発明の第5実施形態に係るコンピュータシステムにおける詰替え処理を説明するためのストライプを示す図。

【図26】同第5実施形態のコンピュータシステムにおける詰替え処理を説明するための論理アドレスタグを示す図。

【図27】同第5実施形態のコンピュータシステムにおける詰替え情報を説明するための図。

【図28】同第5実施形態のコンピュータシステムにお

ける元ストライプ情報を説明するための図。

【図29】同第5実施形態の無効ブロックに対応したビットマップを使った無効ブロック判定のための構成を示す図。

【図30】同第5実施形態のスナップショット採取時に無効ブロックの多いストライプに対するビットマップだけをスナップショット情報として保存する例を示す図。

【図31】この発明の第7実施形態に係るコンピュータシステムのディスク装置をディスクアレイ装置で構成した例を示す図。

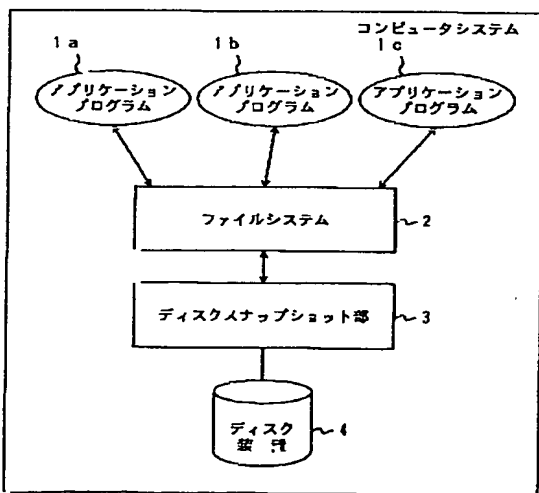
【図32】従来のスナップショットの作成を説明するための図。

【図33】従来のスナップショットの作成の問題点を説明するための図。

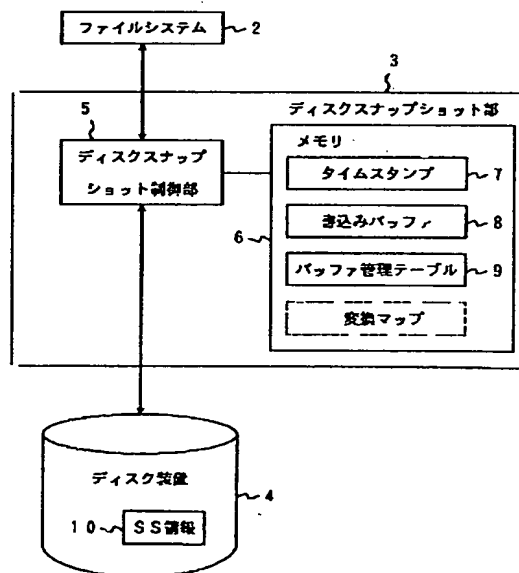
【符号の説明】

1a～1c…アプリケーションプログラム、2…ファイルシステム、3…ディスクスナップショット部、4…ディスク装置、5…ディスクスナップショット制御部、6…揮発性メモリ、7…タイムスタンプ、8…書き込みバッファ、9…バッファ管理テーブル、10…ディスクスナップショット情報、11…スナップショット参照用ファイルシステム、12…スナップショット参照用ユーティリティプログラム、13…スナップショット用ディスク装置。

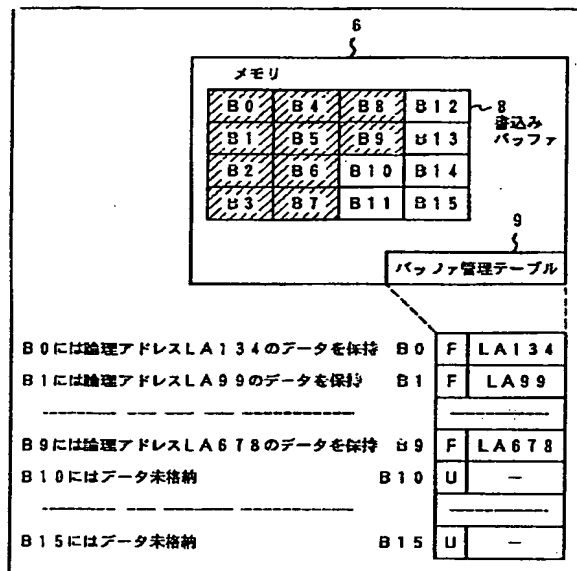
【図1】



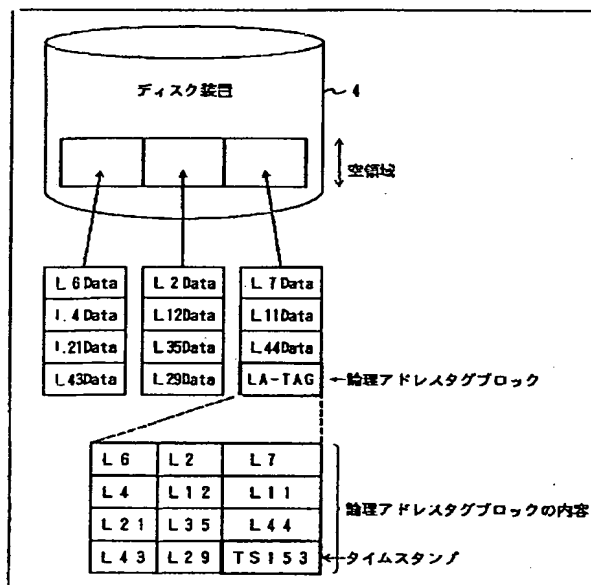
【図2】



【図3】



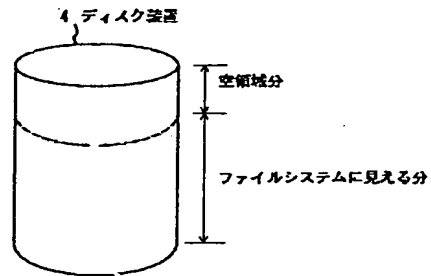
【図5】



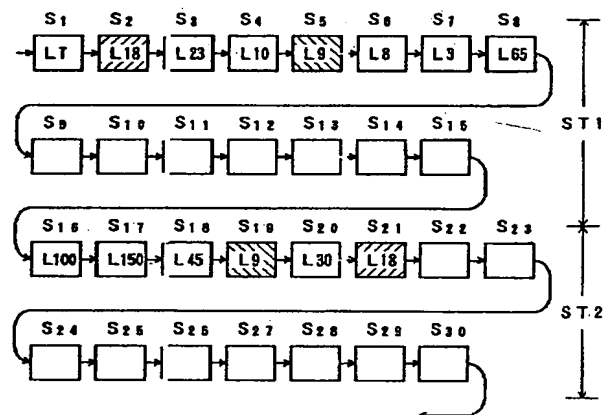
【図8】

論理アドレス	ST#	BLK#	IS#
L0			
L1			
L2			
⋮			

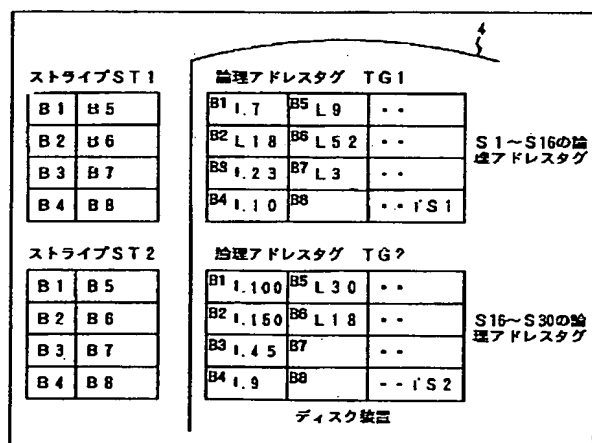
【図4】



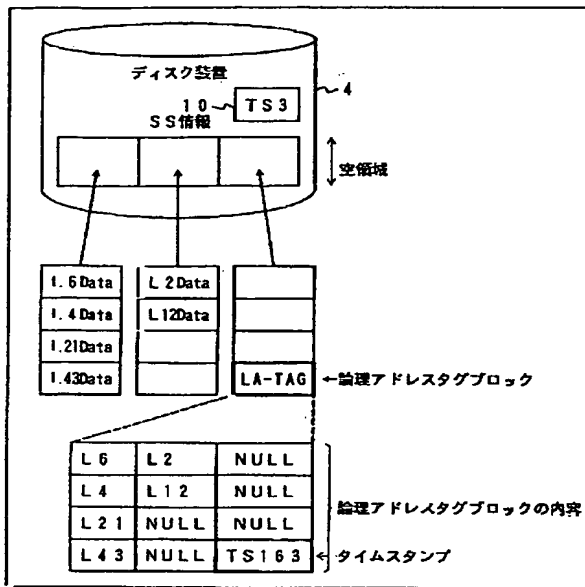
【図6】



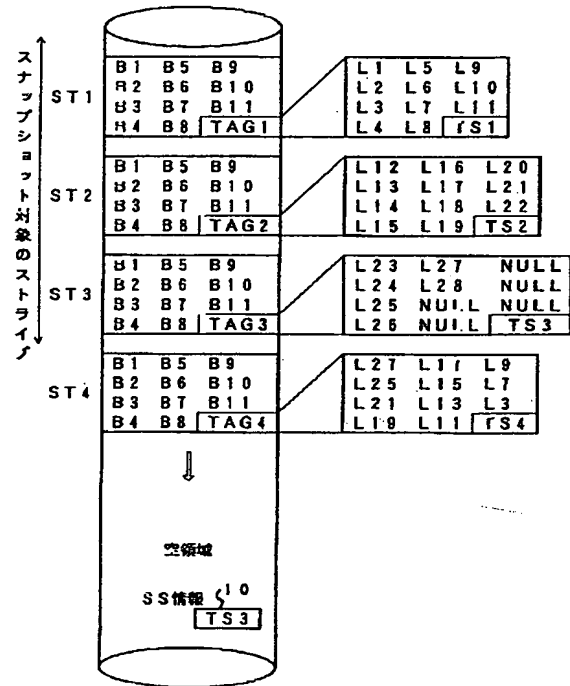
【図7】



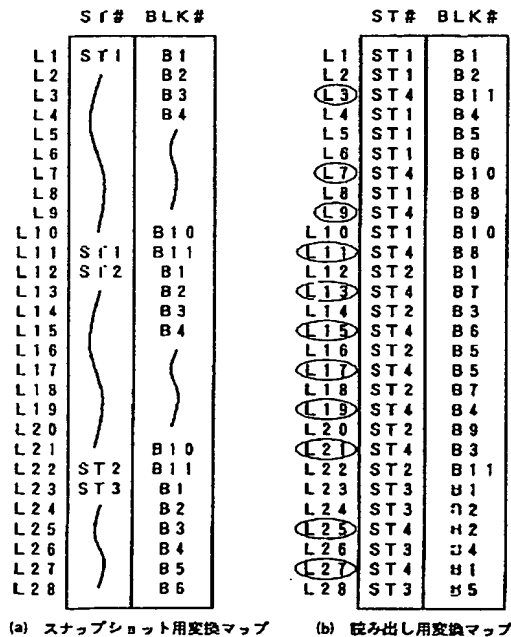
【図9】



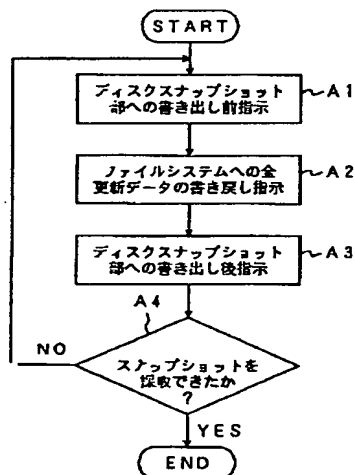
【図10】



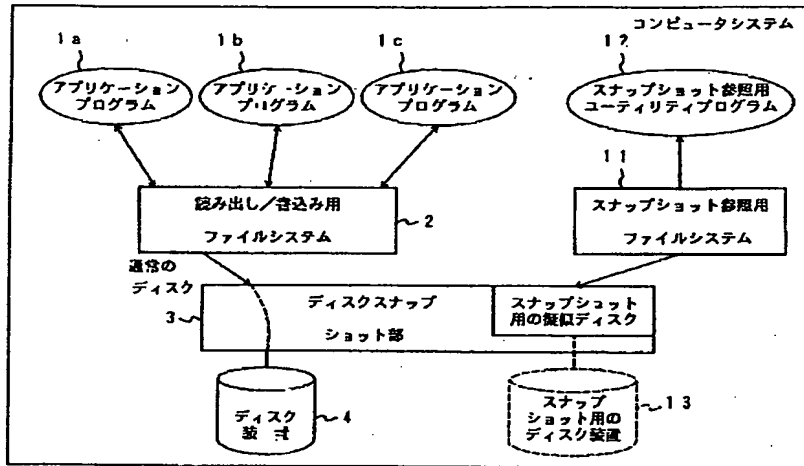
【図11】



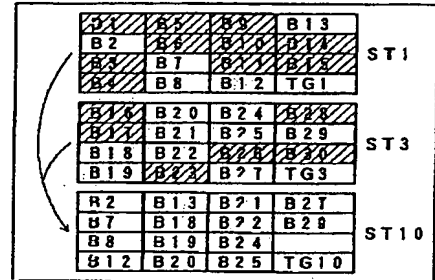
【図13】



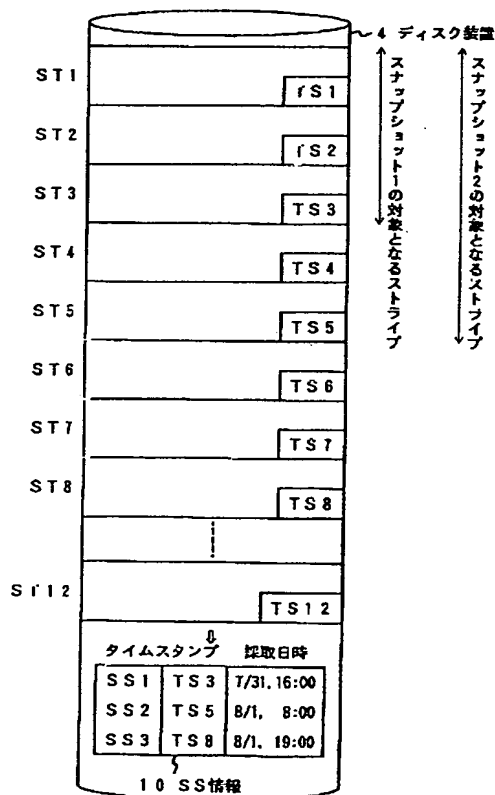
【図12】



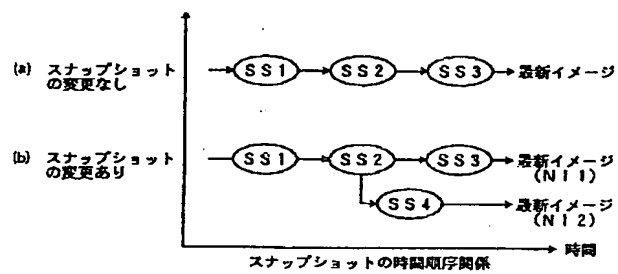
【図25】



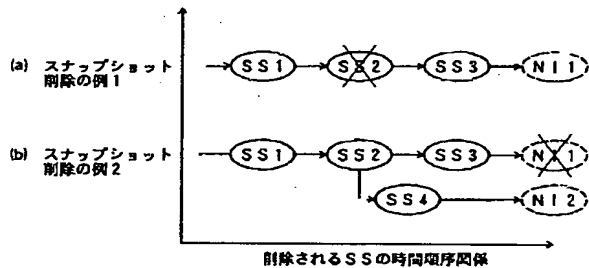
【図14】



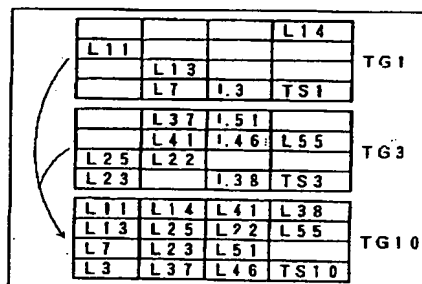
【図20】



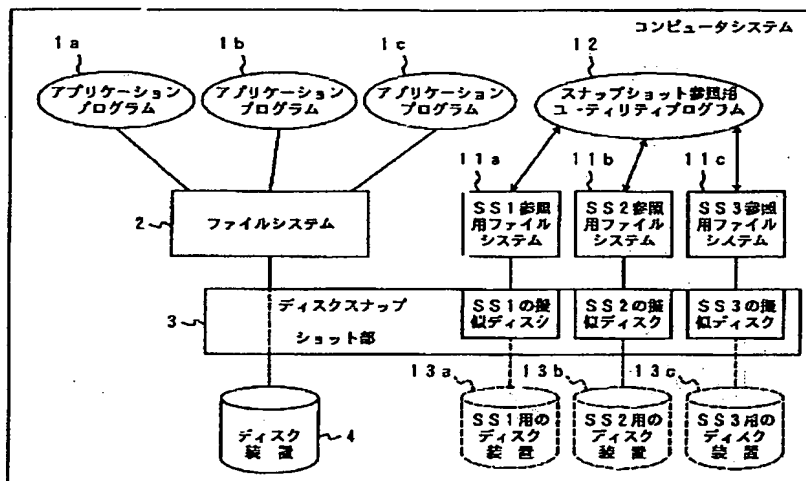
【図22】



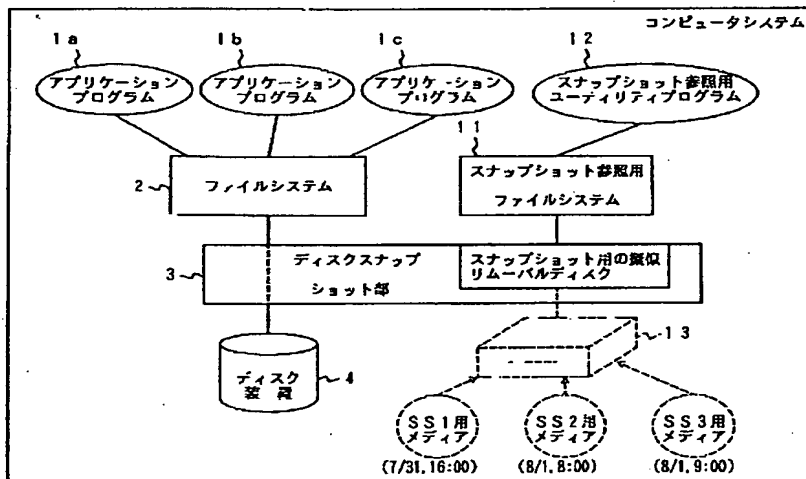
【図26】



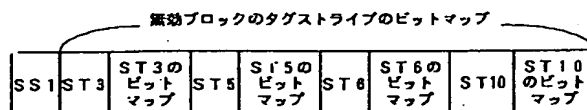
【図15】



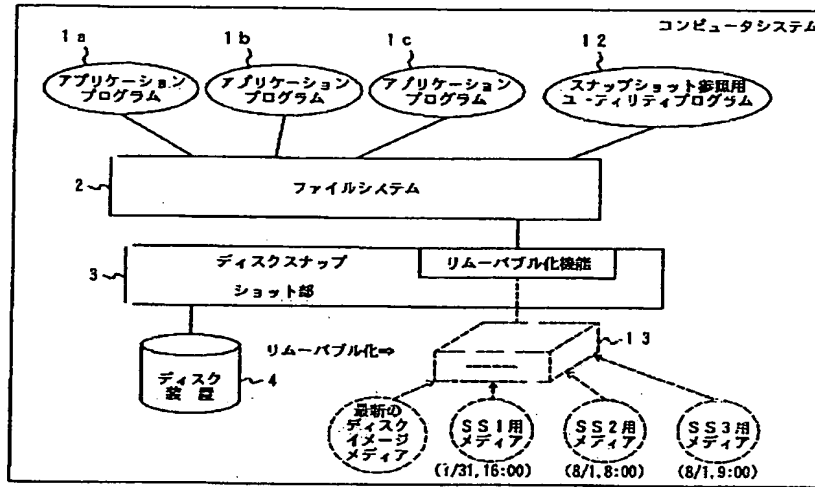
【図16】



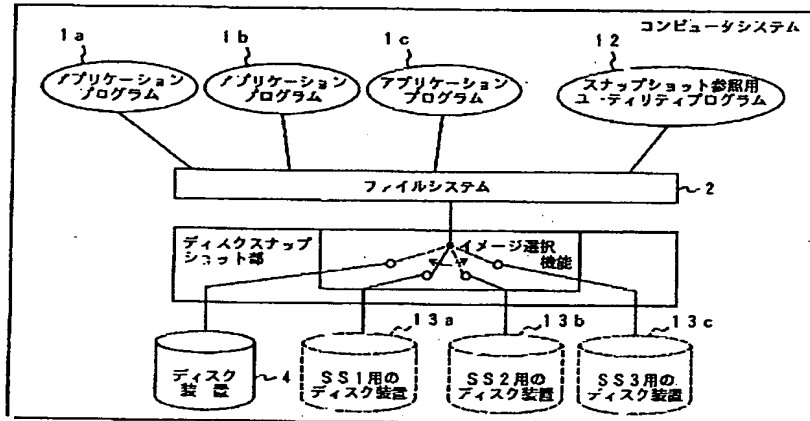
【図30】



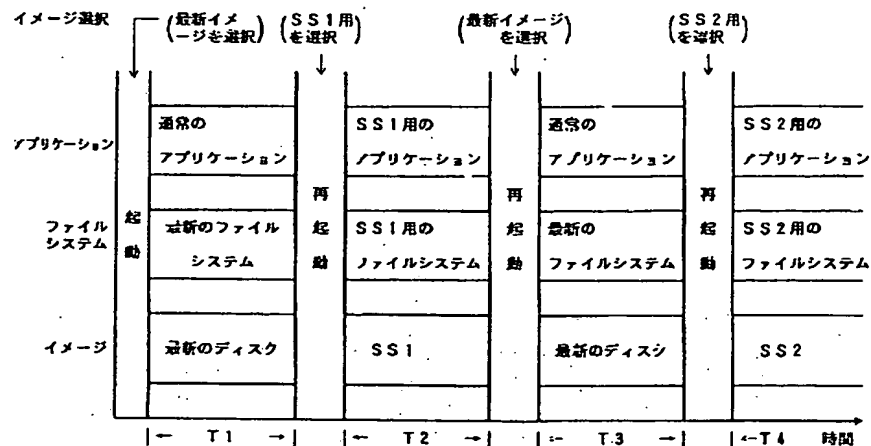
【図17】



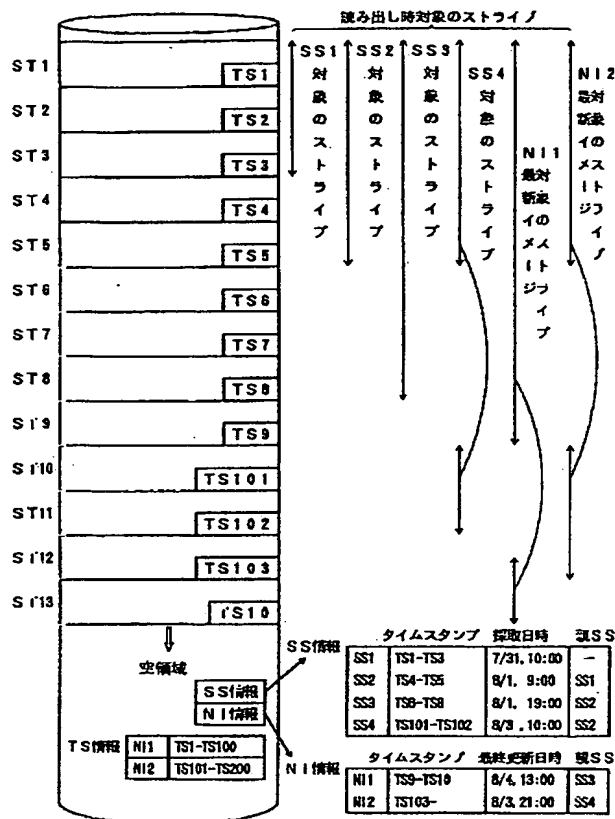
【図18】



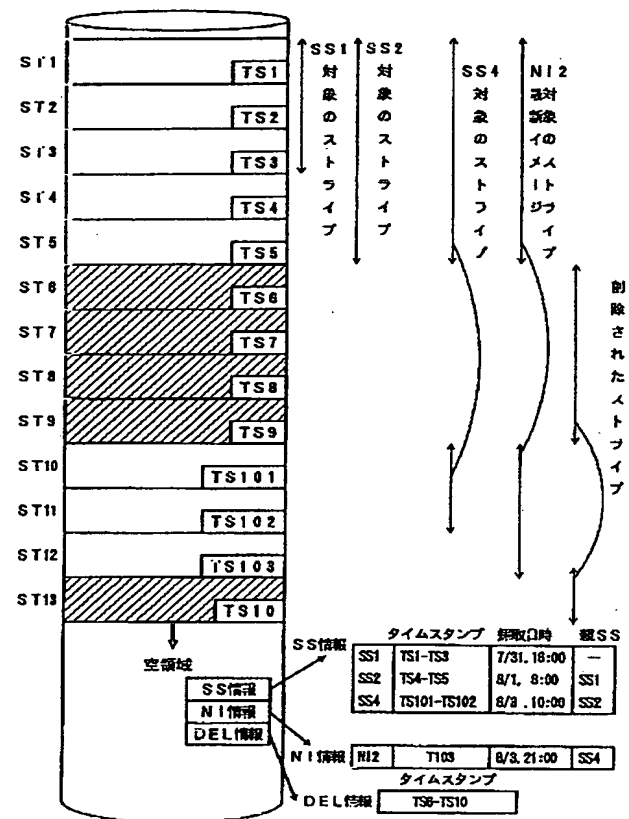
【図19】



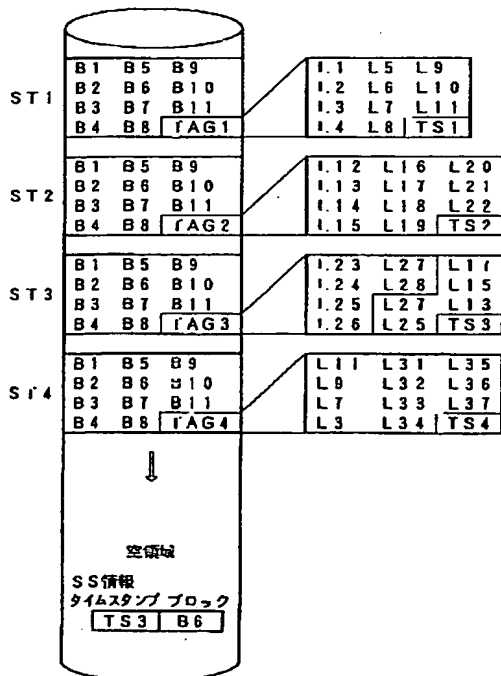
【図21】



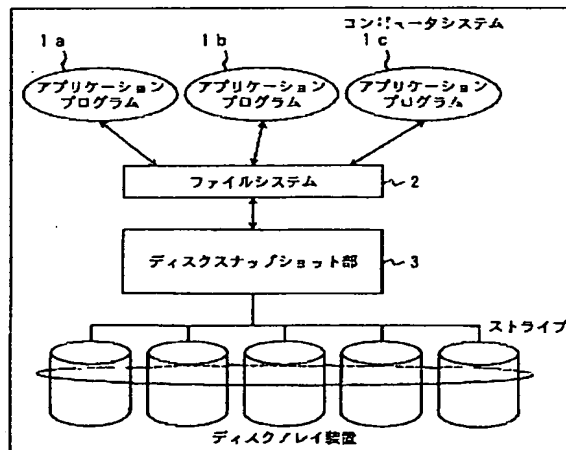
【図23】



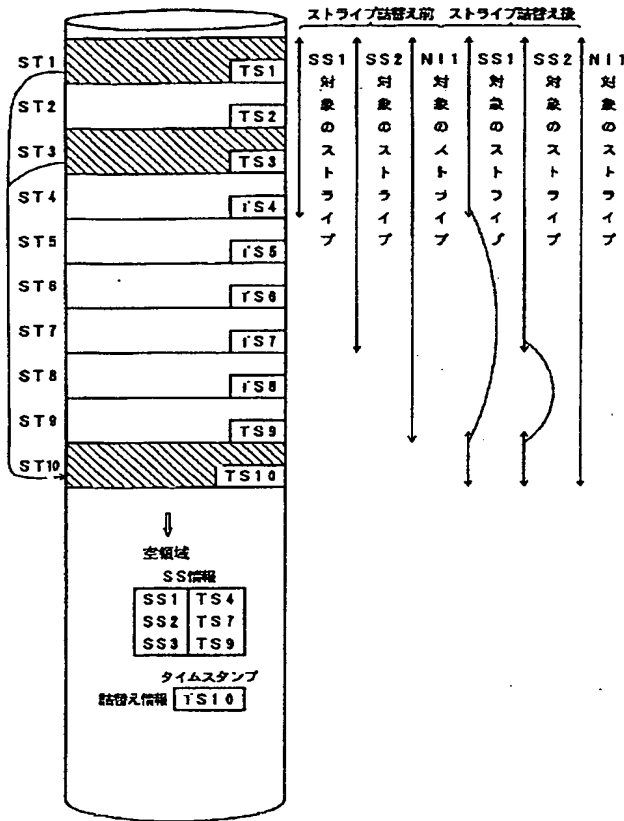
【図24】



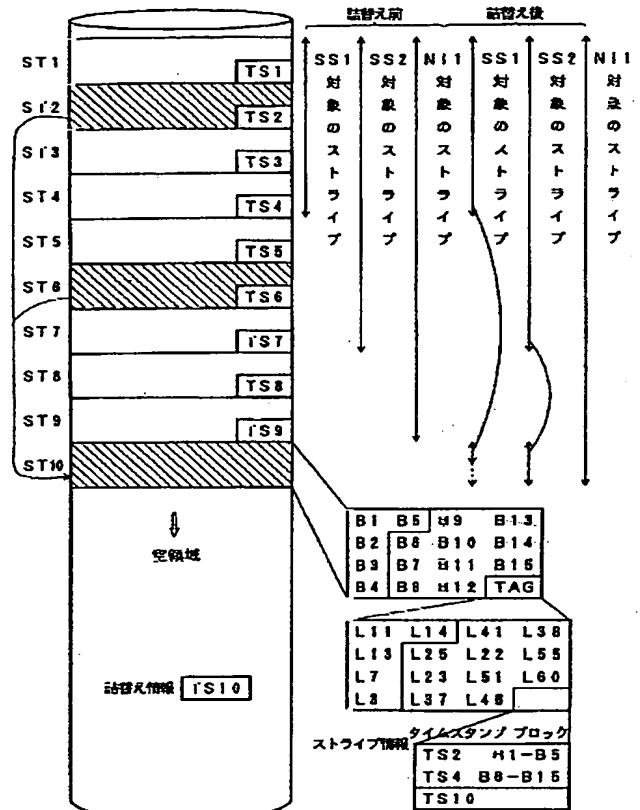
【図31】



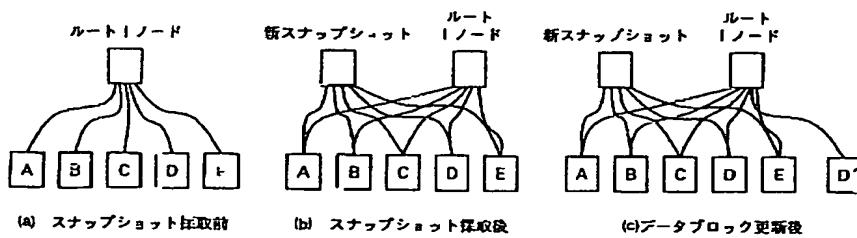
【図27】



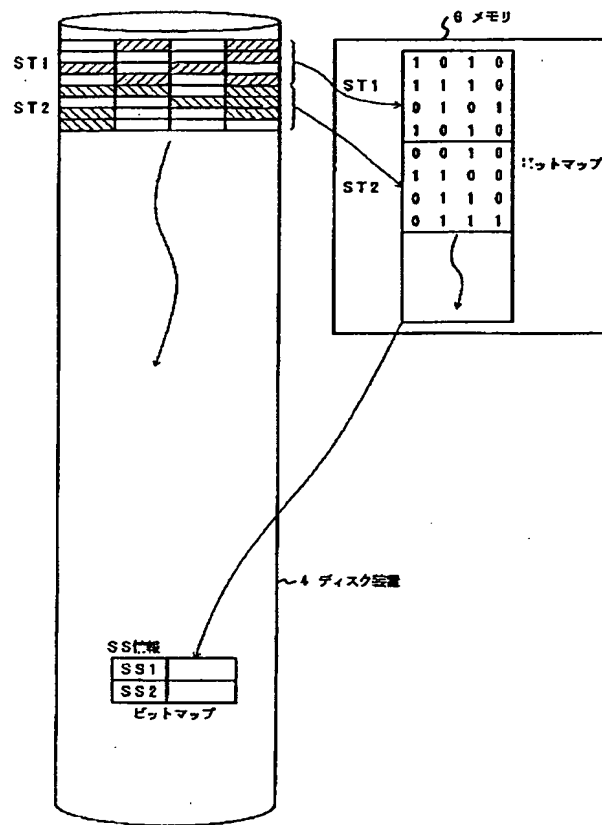
【図28】



【図32】



【図29】



【図33】

